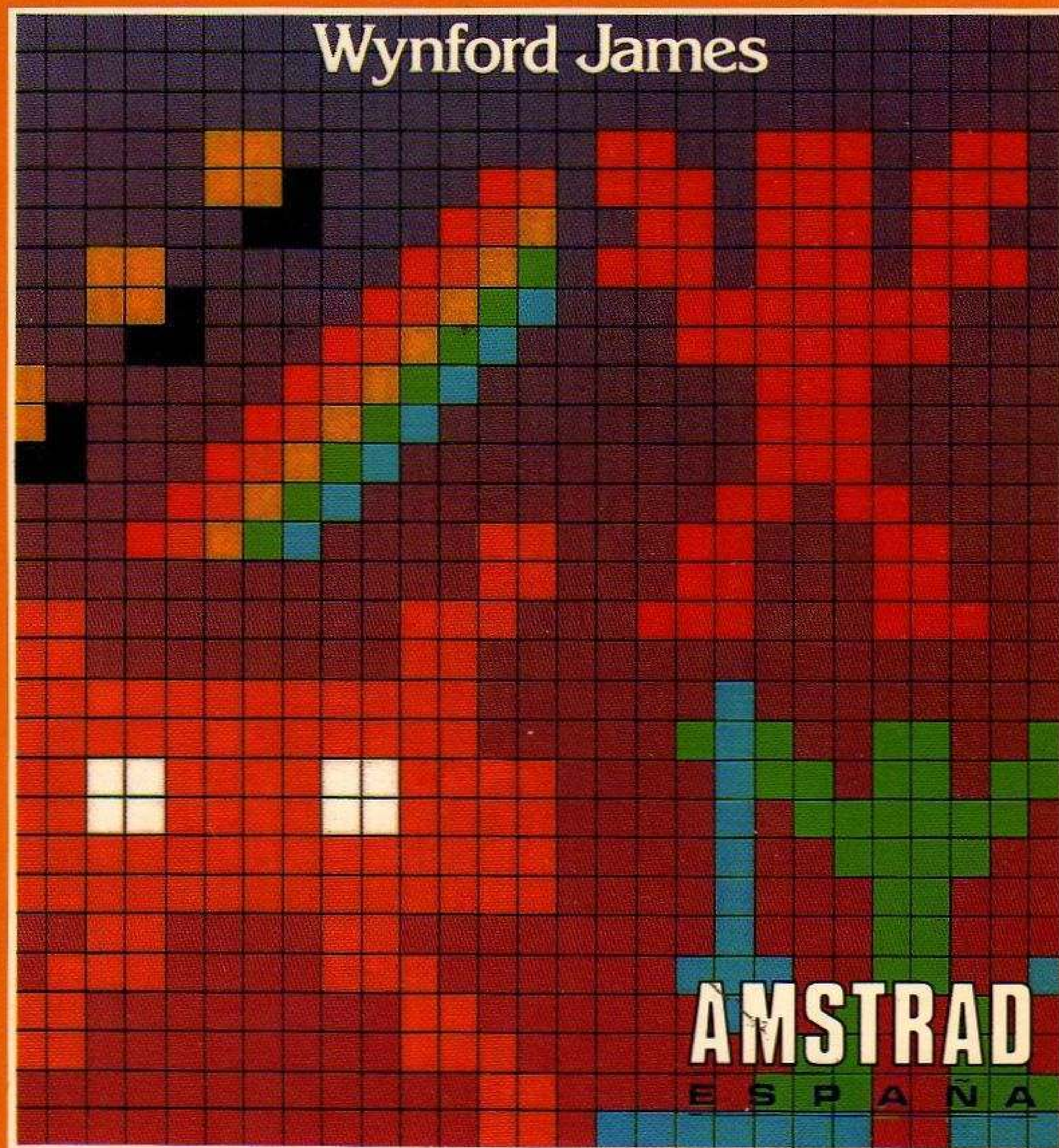
Técnicas de Programación de Gràficos en el Amstrad



TÉCNICAS DE PROGRAMACIÓN DE GRÁFICOS EN EL AMSTRAD

Wynford James



TÉCNICAS DE PROGRAMACIÓN DE GRÁFICOS EN EL AMSTRAD

edición española de la obra
GRAPHICS PROGRAMMING TECHNIQUES ON THE AMSTRAD CPC464
Wynford James

publicada en castellano bajo licencia de
MICRO PRESS
Castle House
27 London Road
Tunbridge Wells, Kent

Traducción
Jesús Rojo García
Profesor de Matemática Aplicada
E. T. S. de Ing. Industriales de Valladolid

INDESCOMP, S.A.
Avda. del Mediterráneo, 9 28007 Madrid

ISBN 84 86176 34 4 Depósito legal: M-26902-85

Impresión:
Gráficas EMA. Miguel Yuste, 27. Madrid
Producción de la edición española:
Vector Ediciones. Gutierre de Cetina, 61. Madrid. (91) 408 52 17

Revisión 2009, por N.:

- Corregidas algunas pequeñas erratas en el texto (ej: incoveniente, fodo), la referencia incorrecta al CPC664 en la página 75, la fórmula final de la rotación antihoraria en 151 y la final en Fig 7.5, las imágenes de las figuras 7.5 y 7.6 estaban intercambiadas.
- Corregidas algunas pequeñas erratas en algún fuente (ej: línea 1409 de la página 47, línea 3110 de la página 73, línea 10072 de la página 144).
- Incluidas capturas de pantalla de la mayoría de los fuentes.
- Sintaxis realzada en colores y con algún sangrado para buscar una mayor legibilidad.

Índice de contenido

Introducción

Este libro pretende enseñarle algunas de las técnicas de programación gráfica que podrá usar en su ordenador Amstrad. Hay capítulos que tratan de animación, con caracteres o con dibujos, de realización de gráficos y de histogramas, de formas geométricas y de otras muchas cosas.

Cada capítulo contiene una serie de programas de ejemplo, muchos de los cuales le serán de gran utilidad. Así, el capítulo 2 contiene un programa que permite diseñar caracteres propios y archivarlos en cinta; el capítulo 3 contiene programas para realizar diagramas de barras y de sectores; el 4 y el 6, programas para la realización de dibujos en la pantalla y su almacenamiento en cinta.

Se suponen conocimientos elementales de BASIC, con algo de manejo de bucles, decisiones condicionales y subrutinas. Aunque los principiantes disfrutarán comprobando los programas escritos en el libro, sacarán mayor provecho si leen antes otro libro del que soy autor: 'Programación Basic con Amstrad', también publicado por Amstrad España.

Para los que ya conocen BASIC pero no han leído mi otro libro, comienzo por resumir en el capítulo 1 lo más interesante de lo que toca a los comandos gráficos del Amstrad.

1. Modos y colores

LA PANTALLA

En la vida real existen distintos tipos de papel para las diferentes actividades. Un arquitecto no dibuja en una libreta los planos de las casas, de la misma manera que no es usual que un novelista escriba sus historias en papel de dibujo. En el ordenador, la pantalla es lo que se utiliza como papel, y es normal que tenga aspecto y textura diferentes cuando se la utiliza para misiones diversas.

El comando **MODE**, seguido del número 0, 1 o 2, sirve para seleccionar uno de los tres aspectos posibles de la pantalla del Amstrad. Cada modo se distingue por el número de caracteres que admite en cada línea, por la cantidad de colores que permite utilizar simultáneamente y por el grado de resolución gráfica (o sea, la finura con la que se pueden dibujar las líneas).

Modo	Número de líneas	Caracteres por línea
0	25	20
1	25	40
2	25	80

Figura 1.1. Los tres modos de la pantalla del Amstrad.

El modo 2 es el ideal para tener en pantalla gran cantidad de texto; en este modo, el Amstrad permite 25 líneas de 80 caracteres cada una.

Cuando se enciende o se inicializa el Amstrad, la pantalla se pone automáticamente en el modo 1. Este modo da 25 líneas de 40 caracteres. El modo 1 proporciona la escritura más legible; es el habitual cuando se están utilizando los comandos del Amstrad.

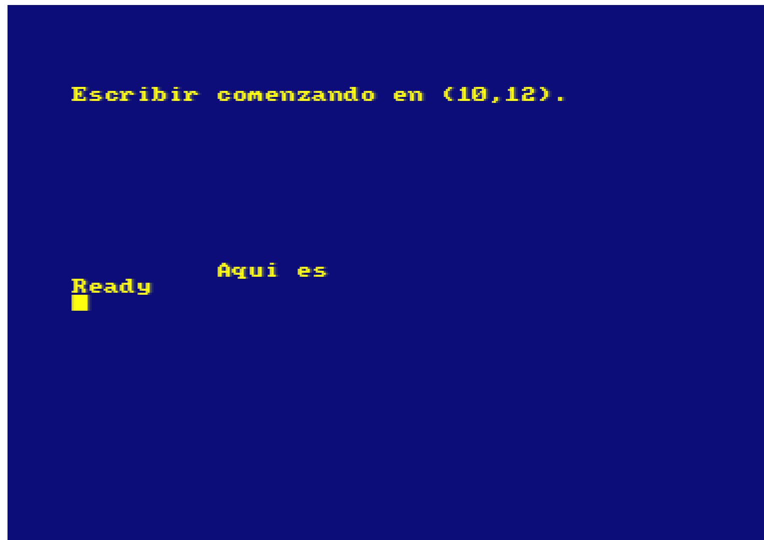
El modo 0 da 25 líneas con sólo 20 caracteres cada una. Es el mejor para realizar figuras en color, ya que admite 16 colores simultáneos.

En cualquiera de los tres modos, se puede escribir en cualquier posición de la pantalla utilizando coordenadas de texto:

```
10 MODE 1
15 PRINT "Escribir comenzando en (10,12)."
20 LOCATE 10,12
30 PRINT "Aqui es"
```

La posición del primer carácter que se escriba será (10, 12). Los números 10 y 12 son las coordenadas de texto de la posición. La primera es la coordenada X (la posición del carácter en la línea) y la segunda la coordenada Y (la línea en la que

está el carácter, empezando a contar desde arriba). La instrucción LOCATE 10,12 es la que hace que comience la escritura en dicha posición. Se puede usar el comando CLS para borrar la pantalla después de ejecutar el programa.



Los números que se pueden utilizar como coordenada X de texto difieren en los diferentes modos de la pantalla, puesto que el número de caracteres por línea es distinto en cada uno de ellos.

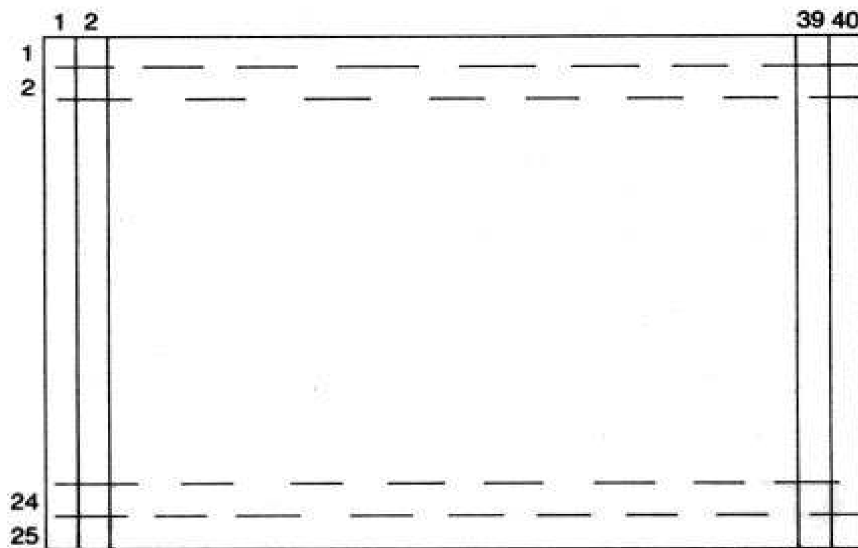


Figura 1.2. La pantalla en modo 1

Si tuviésemos que limitarnos a escribir símbolos en las posiciones de caracteres de la pantalla, las posibilidades de realizar gráficas aceptables serían muy limitadas. Las 25 líneas de 80 caracteres del modo 2 no permiten más que resultados muy pobres a la hora de realizar dibujos a base de caracteres. Afortunadamente, cada posición correspondiente a un carácter puede dividirse en

pequeños trozos; cada uno de ellos es un punto (o pixel).

Dibujando a base de puntos individuales, en lugar de caracteres, se pueden trazar líneas bastante finas. El tamaño de los puntos depende del modo de pantalla, de la misma manera que era variable el número de caracteres por línea y el tamaño de cada carácter.

El sistema de coordenadas de texto no es adecuado para describir la posición de un punto, ya que cada posición de carácter abarca varios puntos. El Amstrad usa un sistema de coordenadas diferente para las posiciones de los puntos; se trata de las coordenadas gráficas.

La pantalla gráfica

El sistema de coordenadas gráficas funciona de manera diferente del de las coordenadas de texto. La pantalla gráfica está dividida horizontalmente en 640 puntos y verticalmente en 400. La posición de cada punto se identifica por su distancia a los bordes izquierdo e inferior de la pantalla.

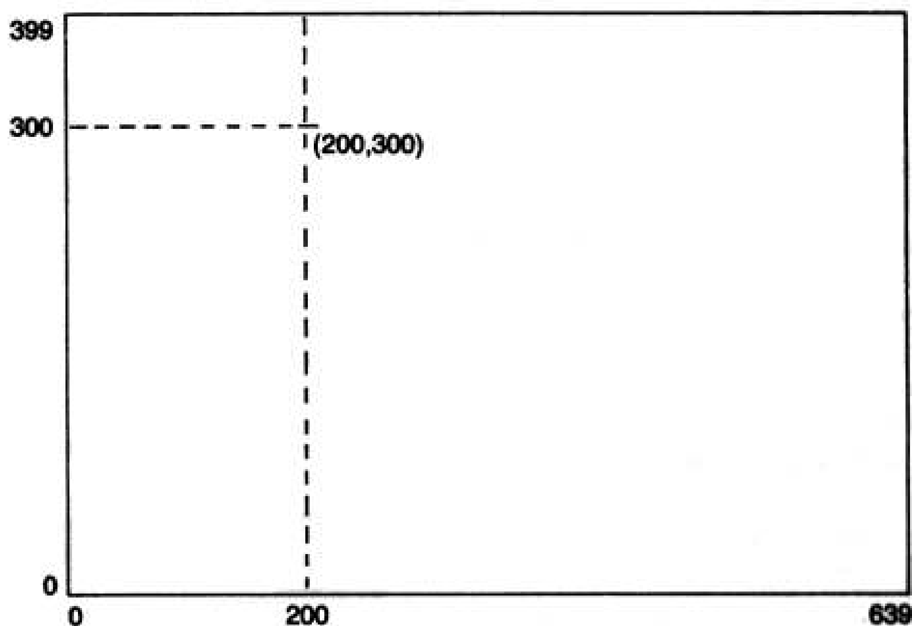


Figura 1.3. La pantalla gráfica; se muestra el punto (200, 300).

La posición del punto de la figura 1.3 es (200, 300). Observe que la segunda coordenada se mide desde el borde inferior de la pantalla, y que el punto de la esquina inferior izquierda de la pantalla tiene coordenadas gráficas (0, 0). Al principio esto puede resultar algo difícil de recordar ya que, para las coordenadas de texto, es el carácter superior izquierdo de la pantalla el que tiene coordenadas (1, 1). Observe además que las coordenadas gráficas comienzan a partir de 0, de manera que el punto de la esquina superior derecha de la pantalla tiene por coordenadas gráficas (639, 399), no (640, 400) como también cabría pensar.

El programa siguiente muestra el uso de dos de los comandos gráficos del

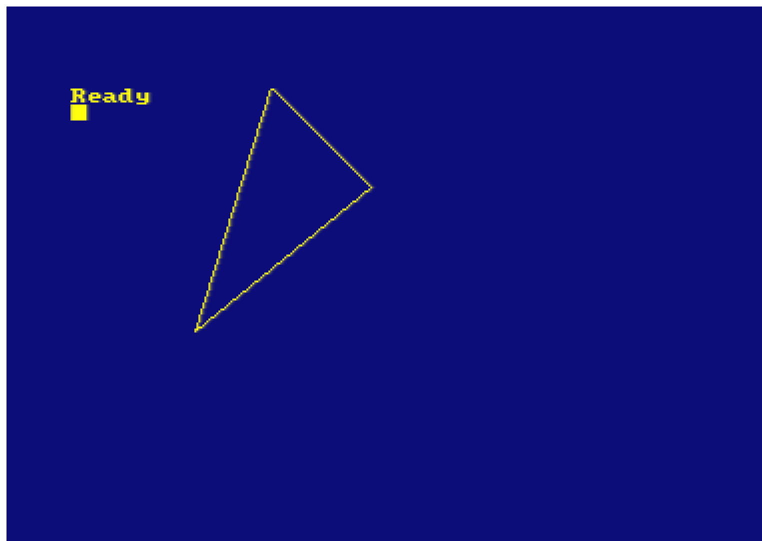
Amstrad:

```
10 MODE 1
20 MOVE 124,156
30 DRAW 300,300
40 DRAW 200,400
50 DRAW 124,156
```

Con él utilizamos el cursor gráfico para dibujar líneas en la pantalla.

Normalmente el cursor gráfico y el de texto permanecen juntos, pero, en cuanto se utiliza un comando gráfico, se está usando el cursor gráfico (invisible).

El comando MOVE de la línea 20 hace que el cursor gráfico se coloque en el punto (124, 156), aunque permanece siempre invisible. El comando DRAW lleva el cursor desde el punto (124, 156) hasta el (300, 300), dibujando una recta entre ambos. Los otros dos DRAW de las líneas 40 y 50 dibujan los dos lados restantes de un triángulo.



Así pues, MOVE x,y lleva el cursor gráfico al punto (x, y) sin dibujar ninguna línea. DRAW x,y, en cambio, dibuja una recta desde el último punto visitado por el cursor gráfico hasta el punto (x, y). Es sencillo dibujar figuras de cierta complejidad almacenando las coordenadas x e y de los puntos en sentencias DATA y haciendo que el programa vaya leyendo estas coordenadas:

```
10 MODE 0
20 x=1
29 REM dibujo a base de MOVE y DRAW
30 WHILE x>0
40 READ x,y
50 READ x1,y1
60 MOVE x,y
```

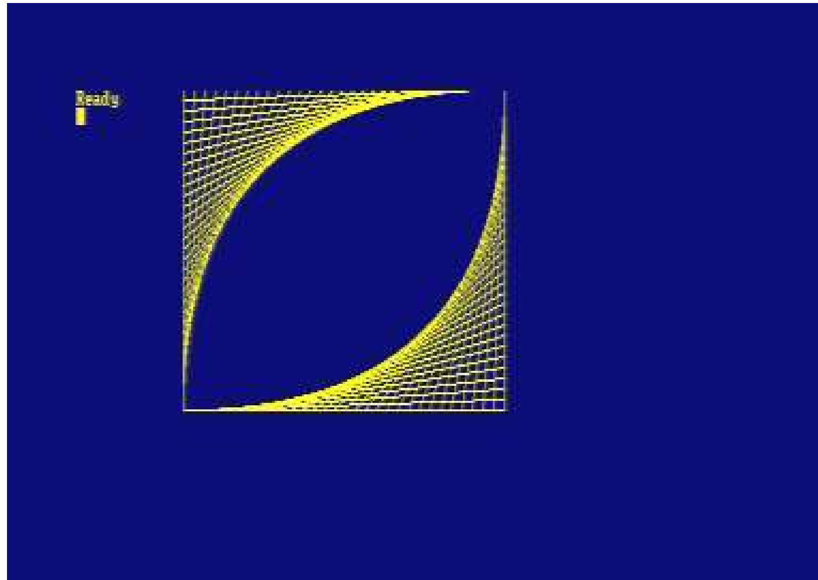
```
70 DRAW x1,y1
80 LOCATE 1,24:PRINT x;" ";y;:r$="":WHILE r$="": r$=INKEY$:WEND
90 WEND
100 END
110 DATA 308,162,344,174,344,174,360,216,360,216,364,260,364,260,360,310
120 DATA 360,310,336,344,336,344,292,356,292,356,248,352,248,352,212,342
130 DATA 212,342,200,316,200,316,196,246,196,246,200,214,200,214,216,174
140 DATA 216,174,252,166,316,198,300,190,300,190,272,190
150 DATA 272,190,256,198,244,200,244,210,244,204,324,204,324,208,324,198
160 DATA 256,238,268,232,268,232,288,232,288,232,296,242,276,242,276,284
170 DATA 288,284,300,296,300,296,324,286,324,286,288,282,264,282,252,294
180 DATA 252,294,232,280,232,280,264,280,196,280,180,296,180,296,172,266
190 DATA 172,266,192,236,364,236,384,266,384,266,380,294,380,294,364,282
200 DATA 320,348,332,324,312,324,312,352,288,354,292,326,272,354,280,326,264,348
210 DATA
224,344,216,334,264,344,268,228,268,218,272,216,272,230,276,228,276,214,280,214,
280,230,280,230,284,212
220 DATA 288,212,284,230,288,228,288,220
230 DATA -1,-1,-1,-1
```



Es posible conseguir efectos gráficos bastante impresionantes con sólo las sentencias MOVE y DRAW. Para conseguir curvas, se utilizan rectas de pequeña longitud empalmadas sucesivamente.

```
1 MODE 2
10 x=100:y=100
20 maximo=300
30 incremento=10
40 FOR numero=0 TO maximo STEP incremento
50   MOVE x+numero,y
60   DRAW x+maximo,y+numero
70   MOVE x,y+numero
80   DRAW x+numero,y+maximo
```

```
90 NEXT
```



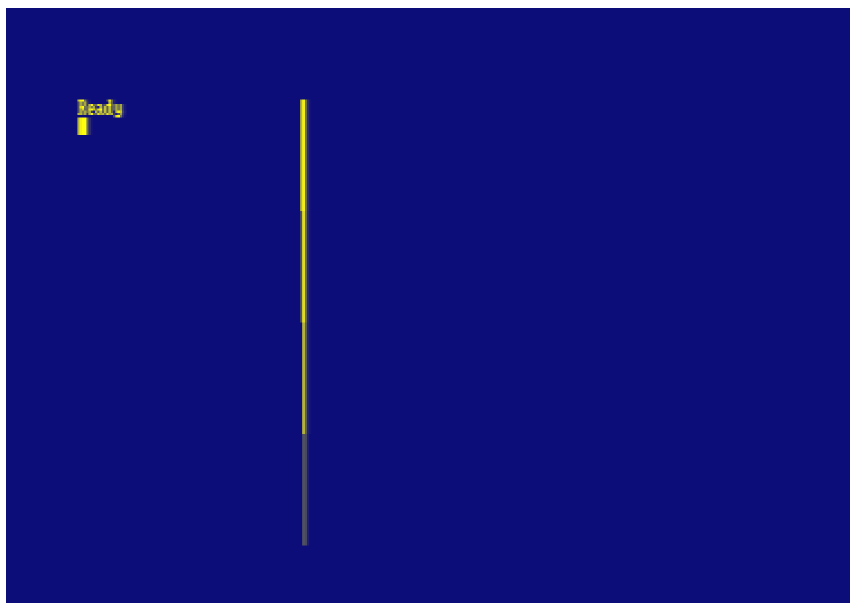
Resolución en los diferentes modos de pantalla

Aunque la pantalla gráfica está dividida horizontalmente en 640 puntos y verticalmente en 400, el Amstrad no puede discriminar realmente todos esos puntos. Además, el poder de discriminación de puntos depende del modo de pantalla, aunque todos los modos utilicen la misma pantalla gráfica. Ejecute el programa anterior cambiando la línea 1 por la siguiente:

```
1 MODE 0
```

El dibujo es el mismo de antes, pero resulta más basto porque las líneas son más gruesas. Pruebe ahora el siguiente programa:

```
10 MODE 2
20 MOVE 200,300
30 DRAW 200,399
40 MOVE 201,200
50 DRAW 201,399
60 MOVE 202,100
70 DRAW 202,399
80 MOVE 203,0
90 DRAW 203,399
```



Las líneas son ahora mucho más finas. El modo 2 es el denominado *modo de alta resolución*, ya que permite distinguir 640 puntos en horizontal y 200 en vertical; por eso resultan tan finas las líneas cuando se utiliza DRAW en este modo.

En el modo 2 no se pueden distinguir en vertical dos puntos consecutivos; los puntos (10, 10) y (10, 11) son exactamente el mismo punto. Los modos 1 y 0 tienen en vertical la misma resolución de 200 puntos que el modo 2, pero bajan mucho en resolución horizontal. Escriba lo siguiente:

```
10 MODE 1
```

y ejecute el programa. El modo 1 es el de *media resolución*, sólo permite distinguir horizontalmente 320 puntos. Esto hace que, en modo 1, los puntos (200, 300) y (201, 300) sean tratados como si fuesen un mismo punto. Escriba ahora:

```
10 MODE 0
```

y ejecute el programa por tercera vez. El modo 0 es el de *baja resolución*, pues sólo puede distinguir 160 puntos en horizontal.

Puede producirle extrañeza el hecho de que alguien elija realizar los dibujos en el modo más basto, cuando es posible la alta resolución del modo 2. La razón es que, aunque el modo 0 da la menor resolución, permite en cambio utilizar 16 colores simultáneos en la pantalla. En este aspecto son inferiores los modos 1 y 2, como muestra la figura 1.4.

Modo	Resolución gráfica	Número de colores simultáneos
0	160 x 200	2
1	320 x 200	4
2	640 x 200	16

Figura 1.4. Resolución gráfica y número de colores disponibles simultáneamente en los diferentes modos de pantalla.

El ordenador tiene una capacidad de memoria limitada: sólo puede almacenar en la RAM una cierta cantidad de información sobre la pantalla. Se trata entonces de llegar a un compromiso, como en tantos otros aspectos de la informática. Se puede utilizar la RAM para guardar información sobre muchos puntos con 2 colores posibles, o de menos puntos con 4 colores, o de muchos menos puntos con 16 colores. El Amstrad le da a elegir; usted debe escoger lo que más le interese en cada caso.

La sentencia PLOT

Las líneas que hemos dibujado en los programas precedentes están formadas por puntos. El Amstrad puede dibujar los puntos uno por uno, si bien en modo 2 es difícil distinguir puntos aislados. De hecho, cada punto está realmente formado por varios puntos físicos de la pantalla; ninguno de los modos es lo suficientemente fino como para identificar un solo punto de pantalla. Lo que nosotros denominamos punto es el menor bloque de puntos de la pantalla que es posible singularizar en cada uno de los modos.

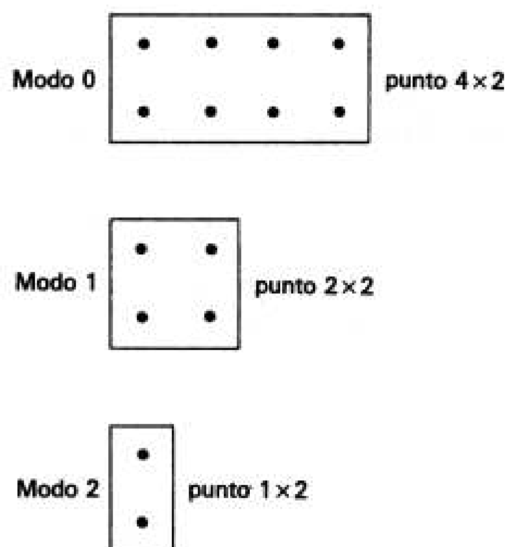


Figura 1.5. Tamaño de los puntos gráficos en cada modo de pantalla

Parece raro permitir que el sistema de coordenadas identifique más puntos de

los que pueden ser dibujados en cualquiera de los modos. La razón principal para ello es que así queda abierta la posibilidad de mejorar en el futuro la resolución gráfica sin que haya que cambiar completamente el sistema de coordenadas.

La instrucción PLOT sirve para dibujar un punto aislado. Funciona de la misma manera que MOVE o DRAW; la palabra PLOT debe ir seguida de las coordenadas gráficas del punto que se desea dibujar. El siguiente programa dibuja seis puntos distintos:

```
10 MODE 0
20 PLOT 160,200
30 PLOT 320,200
40 PLOT 324,200
50 PLOT 328,200
60 PLOT 332,200
70 PLOT 480,200
```

entre las líneas 30 si 60 puedan unidos formando un pequeño segmento de recta. En modo 1 los mismos puntos quedan separados. En modo 2, los puntos son tan pequeños que le resultará difícil distinguirlos.

LOS COLORES

Al encender el Amstrad, el texto y los gráficos aparecen en color amarillo sobre fondo azul, independientemente del modo en el que se trabaje. En realidad, se dispone de 27 colores diferentes, no todos ellos perfectamente distinguibles unos de otros. Cada color tiene asignado un número; para referirse a un color se utiliza siempre ese número en lugar del nombre del color.

Número	Color
0	Negro
1	Azul
2	Azul intenso
3	Rojo
4	Magenta (rojo + azul)
5	Malva
6	Rojo intenso
7	Morado
8	Magenta intenso
9	Verde
10	Cyan
11	Azul celeste

12	Amarillo
13	Blanco
14	Azul pastel
15	Naranja
16	Rosa
17	Magenta pastel
18	Verde intenso
19	Verde marino
20	Cyan intenso
21	Verde lima
22	Verde pastel
23	Cyan pastel
24	Amarillo intenso (dorado)
25	Amarillo pastel
26	Blanco intenso

Figura 1.6. Los 27 colores del Amstrad con sus números de identificación.

Conviene ahora advertir que el Amstrad no trabaja sobre toda la pantalla, sino sobre un gran rectángulo de la misma. Este rectángulo está rodeado por un borde que no se utiliza. Al encender el ordenador, el borde está del mismo color que el fondo de la pantalla. Este borde no forma parte de la memoria del ordenador, ya que no se escribe ni se dibuja nunca en él.

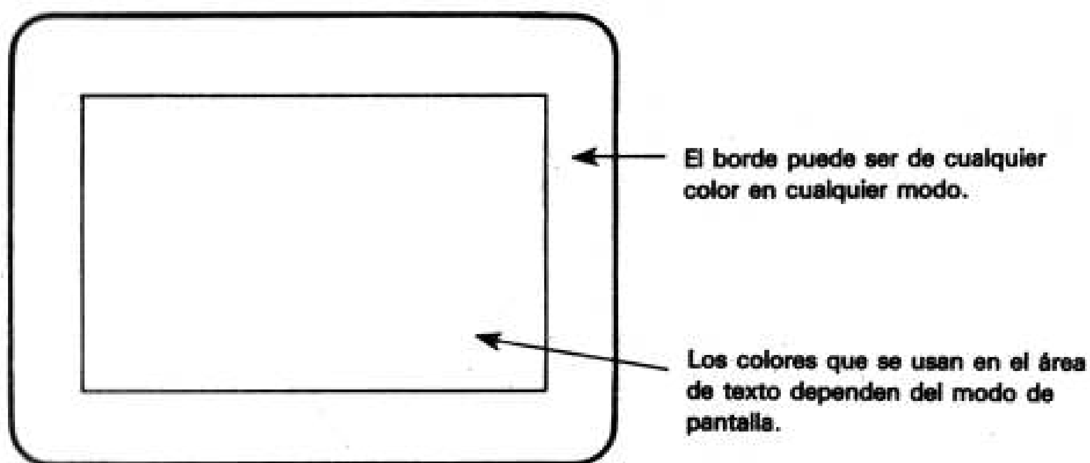


Figura 1.7. El borde de la pantalla en su monitor o televisor.

El borde puede tener cualquier color en cualquier modo. No existen restricciones sobre el color que se emplee para el borde. El modo 2 sólo puede utilizar dos colores en la pantalla, pero en cambio el borde puede tomar cualquier otro color (de los 27 disponibles). Ejecute las instrucciones:

```
10 MODE 2
20 BORDER 0
```

Según la figura 1.6 el 0 es el número del color negro. Al ejecutar BORDER 0 hemos teñido de negro el borde de la pantalla. Se puede cambiar el 0 por otros números, entre 0 y 26, para probar el efecto de distintos colores del borde. Por ejemplo, BORDER 26 pone el borde de blanco.

En los modos 0 y 1 el borde funciona de la misma manera. Como se puede observar, el cambio de modo de pantalla no modifica el color del borde. Al encender el ordenador el borde se pone automáticamente de color azul, como con BORDER 1.

PEN y PAPER

También se puede cambiar los colores que se utilizan en el rectángulo principal de la pantalla. Pero aquí interviene la cantidad de memoria que se utiliza en el almacenamiento de la pantalla a la hora de restringir el número de colores que pueden ser utilizados simultáneamente.

Vamos a ver cuál es la situación al encender el ordenador. Más adelante veremos cómo se puede cambiar esta situación con el comando INK.

Comenzaremos por hacer algún ensayo. Escriba MODE 0 y PEN 4. Como "pen" significa pluma y el 4 es el número del color magenta, cabría esperar que el texto fuese en adelante de este color. Sin embargo, todo lo que se escriba de ahora en adelante será de color blanco. Si se teclea PEN 5, la pluma comienza a escribir en negro. Finalmente, si se teclea PEN 14 el texto que se escriba quedará parpadeando entre los colores amarillo y azul.

Lo que ocurre es que en modo 0 hay 16 tintas disponibles; la instrucción PEN lo que hace es cargar la pluma con una de esas tintas. La figura 1.8 indica de qué colores son inicialmente las 16 tintas en el momento de encender el ordenador.

Número de PEN o PAPER	Modo 0	Modo 1	Modo 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	1/24	20	1
15	16/11	6	24

Figura 1.8. Los colores de las tintas al encender el ordenador. Un color doble (tintas 14 y 15 del modo 0) indica una tinta parpadeante entre los dos colores indicados.

Además de la pluma, también se puede cambiar el color del fondo con la instrucción PAPER (o sea, papel). Esta instrucción asigna para el fondo una tinta determinada. Comience por inicializar el ordenador mediante las teclas <CTRL> <SHIFT> <ESC> y teclee MODE 0. Teclee luego PAPER 3. Lo que escriba a continuación aparecerá sobre fondo rojo, que es el color de la tinta número 3 (el color 6). Si quiere que cambie todo el fondo de la pantalla a este color, utilice el comando CLS. PAPER permite en modo 0 utilizar para el fondo cualquiera de las 16 tintas posibles, incluidas las parpadeantes.

Ahora podemos combinar las dos sentencias que hemos visto para seleccionar los colores del texto y el fondo. Por ejemplo,

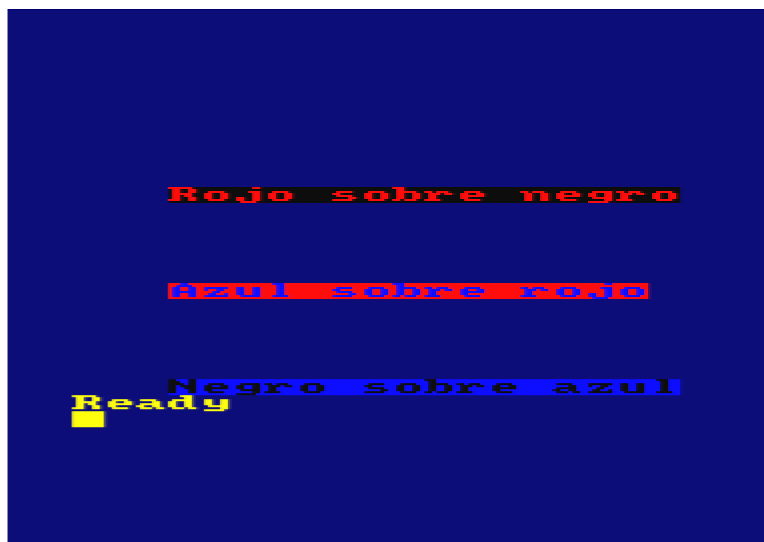
```
PEN 3
PAPER 4
CLS
```

selecciona caracteres rojos sobre fondo blanco en el modo 0.

Con los modos 1 y 2 todo ocurre de forma parecida, pero se pueden seleccionar menos tintas. En modo 1 las 16 tintas se reducen en la práctica a 4, y en el modo 2 se reducen a dos. En el modo 2, ocho tintas son del color amarillo y otras ocho del azul, aunque estos colores se pueden cambiar por otros dos cualesquiera. Conviene que observe en la figura 1.8 que una misma tinta puede tener colores diferentes en los distintos modos de pantalla. La principal consecuencia de este hecho es que un programa que trabaje perfectamente en el modo 0 puede no funcionar cuando se lo ejecuta en modo 2, si las tintas de la pluma y del fondo producen un mismo color en este último modo.

Naturalmente, los comandos PEN y PAPER se pueden utilizar también incluidas en líneas de programa, como por ejemplo:

```
10 MODE 0
20 LOCATE 4,7
30 PEN 3
40 PAPER 5
50 PRINT "Rojo sobre negro"
60 LOCATE 4,13
70 PEN 6
80 PAPER 3
90 PRINT "Azul sobre rojo"
100 LOCATE 4,19
110 PEN 5
120 PAPER 6
130 PRINT "Negro sobre azul"
140 REM valores normales de PEN y PAPER
150 PEN 1
160 PAPER 0
```



Puede ejecutar este programa en los tres modos, cambiando para ello la línea 10. Los resultados que se obtienen pueden ser curiosos, ya que las tintas son de colores diferentes en los distintos modos.

He aquí otro ejemplo:

```
10 MODE 0
20 rojoenmodo0=3
30 negroenmodo0=5
40 PEN rojoenmodo0
50 PAPER negroenmodo0
60 CLS
70 LOCATE 8,12
80 PRINT "Vale!"
90 REM valores normales de PEN y PAPER
100 PEN 1
110 PAPER 0
```

Las dos últimas líneas se encargan de restablecer las tintas del fondo y la pluma a sus valores habituales, para evitar encontrarse al terminar el programa con cosas como texto amarillo sobre fondo blanco, o con cualquier otra mezcla que haga ilegible el texto.



Precisamente, uno de los problemas habituales cuando se trabaja con los colores es que, al terminar un programa, podemos encontrarnos con que el color de la pluma y del fondo coinciden. Una forma de solucionar esto es proceder como en los programas anteriores. Otra alternativa es programar una de las teclas de función para que restituya los colores habituales; por ejemplo,

```
KEY 128,CHR$(13)+"INK 0,1:INK 1,24:PEN 1: PAPER 0"+CHR$(13)
```

Ejercicios

1.Escriba un programa que elija aleatoriamente un punto de la pantalla y trace una línea hasta otro punto, elegido también aleatoriamente, y que repita el proceso hasta un nuevo punto, y así hasta dibujar 100 líneas.

2.Dibuje en la pantalla un cohete utilizando las sentencias MOVE y DRAW. Escriba en el cohete el nombre que usted quiera darle.

3.Escriba bien centrada en la pantalla del modo 0 la frase "colores surtidos", haciendo que cada letra sea de un color diferente.

Gráficos y colores

Es fácil en el Amstrad realizar dibujos a base de líneas de colores. Si se utilizan los comandos MOVE y DRAW como hemos hecho hasta ahora, se obtienen líneas del color de la tinta 1. Como esta tinta era del color 24 en todos los modos, hemos obtenido siempre líneas de color amarillo intenso.

Para trazar líneas de otros colores se debe emplear una generalización del comando DRAW. Reinicialice el ordenador y escriba:

```
MOVE 100,100
DRAW 300,300,2
```

Obtendrá una línea que une los puntos (100, 100) y (300, 300), dibujada con el color de la tinta 2, que es (en modo 1) de color cyan intenso, el número 20. Escriba ahora:

```
MOVE 300,300
DRAW 400,0,3
```

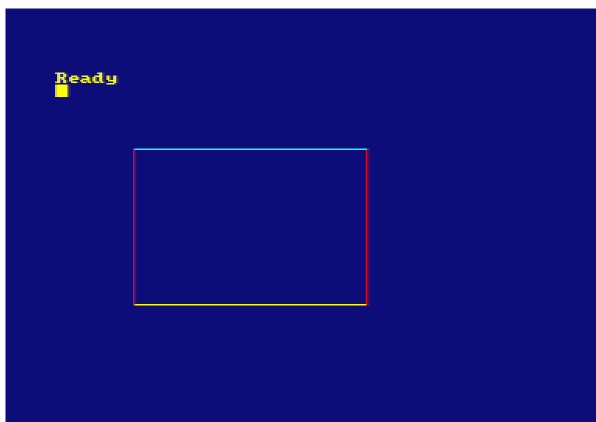
y obtendrá entre (300, 300) y (400, 0) una línea roja, puesto que la tinta 3 es del color número 6 en modo 1.

Con la misma facilidad se pueden utilizar estos comandos incluidos en los programas.

Recuerde que ya explicamos que un programa que funciona correctamente en un modo de pantalla, puede no hacerlo en otro a causa del diferente significado de las tintas en los distintos modos. El siguiente programa dibuja un rectángulo en modo 1, con un lado amarillo, otro cyan y los otros dos rojos:

```
10 MODE 1
20 MOVE 100,100
30 DRAW 400,100
```

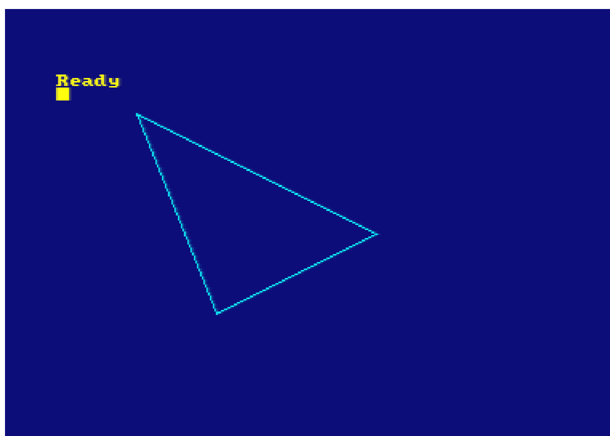
```
40 DRAW 400,300,3
50 DRAW 100,300,2
60 DRAW 100,100,3
```



En la línea 30 no se especifica el número de la tinta, por lo que el ordenador utiliza la número 1. Pero, ejecute el programa una segunda vez. Le sorprenderá seguramente la desaparición del lado amarillo.

Cuando el ordenador encuentra un comando DRAW sin número de tinta, lo que hace es conservar el último que se haya utilizado. Al ejecutar por segunda vez el programa, la última tinta utilizada es la número 3, que será la que utilice entonces en el DRAW de la línea 30. Esto tiene la ventaja de que no es necesario especificar un nuevo color mientras se desee seguir utilizando el mismo. Compruebe este hecho con el programa:

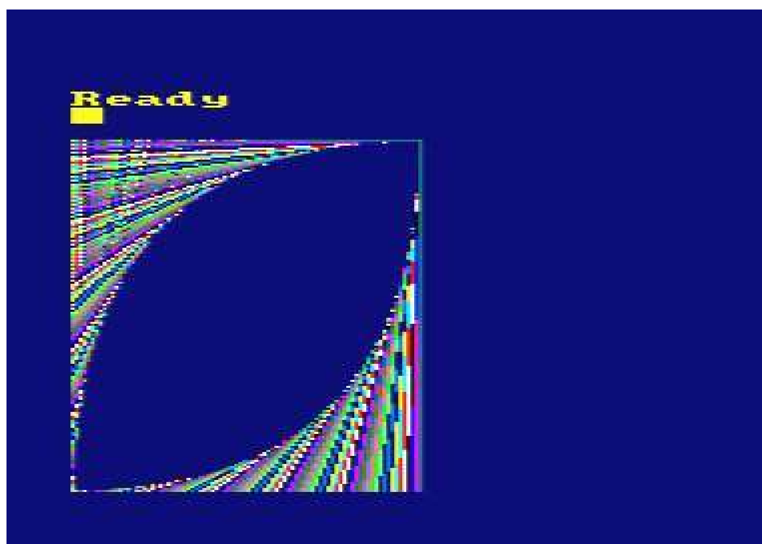
```
10 MODE 1
20 MOVE 200,100
30 DRAW 400,200,2
40 DRAW 100,350
50 DRAW 200,100
```



Pruebe también este programa en el modo 2, donde la tinta número 2 es de un color diferente.

El modo 0 es con mucho el mejor para dibujar figuras con gran colorido, siempre que no importe excesivamente la resolución:

```
10 MODE 0
20 x=0:y=0
30 color=1
40 FOR con=0 TO 350 STEP 4
50 MOVE x+con,y
60 DRAW x+350,y+con,color
70 MOVE x,y+con
80 DRAW x+con,y+350
90 color=(color+1) MOD 16
100 NEXT
```



Cómo cambiar las tintas

Hasta ahora sólo hemos utilizado unos cuantos de los colores que puede producir el Amstrad. Hay 16 tintas utilizables, pero hay 27 colores, como vimos en la figura 1.6. El Amstrad permite cambiar los colores de las tintas, de manera que podemos así seleccionar en un modo particular la gama de colores que interese.

Lo que no se puede alterar es la cantidad de colores que pueden aparecer simultáneamente en la pantalla, pues es una cantidad fija que depende del modo de pantalla. Son siempre 2 colores en modo 2, 4 colores en modo 1 y 16 colores en modo 0. Por ejemplo, se puede lograr que en modo 2 el texto sea rojo y el fondo blanco, pero solo estos dos colores podrán estar en la pantalla en un instante dado.

Al encenderlo, el ordenador se pone en modo 1, el fondo queda de la tinta 0 (PAPER 0), que es de color 1, y el texto de la tinta 1 (PEN 1), que es de color 24. Reinicialice el ordenador y escriba

```
INK 1,6
```

Todo el texto que hay en la pantalla cambiará instantáneamente a color rojo intenso. El comando INK (o sea, "tinta") ha de ir seguido de dos números. El primero es el número de la tinta cuyo color se va a establecer; el segundo es el número del color que se asigna a la tinta.

El comando INK 1,6 ha servido para cambiar la tinta 1 al color 6, que es rojo intenso. Todo lo que esté escrito o dibujado con la tinta 1 cambiará automáticamente al nuevo color. Para poner el texto en azul utilizaremos

```
INK 1,2
```

Lo que estaba en rojo se cambia a azul intenso. ¿Desea volver el texto al color usual? Posiblemente sepa ya como hacerlo. Escriba

```
INK 1,24
```

Normalmente, la tinta 1 es en todos los modos del color 24.

Los cambios de color de las tintas sirven también para cambiar el color del fondo. En este momento el fondo es de la tinta 0 (PAPER 0), de color azul.

Para cambiarlo a blanco, escriba

```
INK 1,26
```

Seguramente le será difícil leer el texto. Pruebe con INK 0,6 o con INK 0,0. Normalmente el fondo es azul en todos los modos, del color 1. Pruebe a volver el fondo de dicho color.

No es necesario haber utilizado previamente una tinta para cambiarle el color. Inicialice el ordenador y escriba

```
INK 3,0
```

No parece que haya pasado nada. Si ahora elegimos PEN 3 en el modo 1, la tabla de la figura 1.8 parece sugerir que obtendremos textos de color rojo intenso.

Pero lo que hemos hecho ha sido cambiar la tinta 3 del color habitual al color O, que es negro. Si escribe

```
PEN 3
```

verá que el nuevo texto aparecerá en negro. Escriba ahora

```
INK 3,6
```

y todo el texto escrito con PEN 3 pasará a rojo intenso. Este color permanece incluso si se cambia de modo; compruébelo.

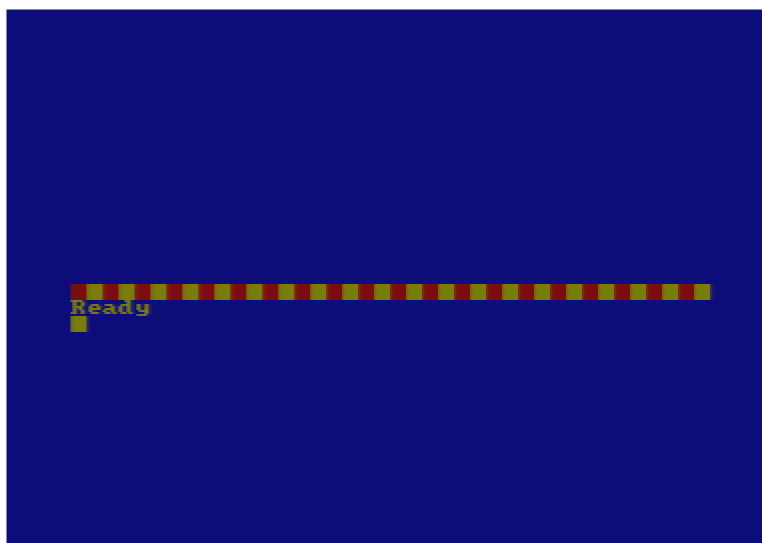
Es posible también crear tintas que sean intermitentes entre dos colores. Escriba

```
INK 1,3,26
```

y todo el texto que se haya escrito con PEN 1 comenzará a alternar entre los colores 3, que es rojo, y 26, que es blanco.

La adecuada utilización de tintas intermitentes puede servir para dar la impresión de movimiento. Si se hace que la tinta 1 sea intermitente amarillo/rojo y la tinta 2 Io sea rojo/amarillo, y se escriben caracteres alternos de las dos tintas, se produce la ilusión de un desplazamiento de los colores a lo largo de la línea:

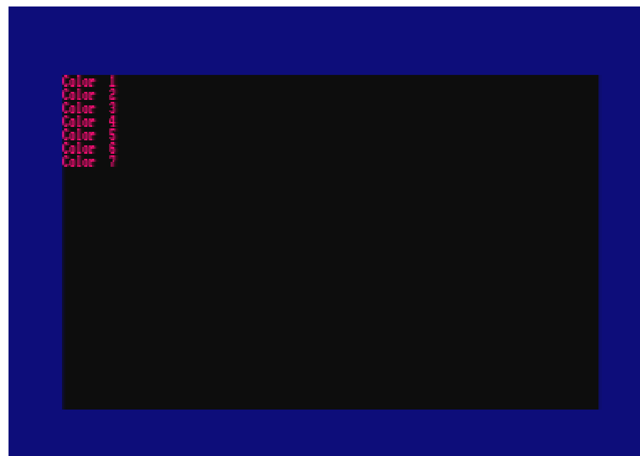
```
10 MODE 1
20 INK 1,3,12
30 INK 2,12,3
40 tintapluma=1
50 FOR x=1 TO 40
60 IF tintapluma=1 THEN tintapluma=2 ELSE tintapluma=1
70 PEN tintapluma
80 LOCATE x,13
90 PRINT CHR$(143);
100 NEXT
```



La indudable ventaja del comando INK es que permite seleccionar cualquier combinación de los 27 colores. Incluso en el modo 2, que sólo admite dos tintas, se pueden conseguir buenos efectos usando combinaciones tales como texto rojo sobre fondo blanco; es decir, no estamos limitados a la combinación de amarillo y

azul que se obtiene al encender el ordenador. El siguiente programa muestra las más de 700 combinaciones de colores posibles en el modo 2:

```
10 MODE 2
20 FOR x=0 TO 26
30   CLS
40   INK 0,x
50   FOR y=0 TO 26
60     IF x<>y THEN INK 1,y:PRINT "Color "; y
70     respuesta$=""
80      WHILE respuesta$=""
90       respuesta$=INKEY$
100      WEND
110    NEXT
120 NEXT
```



Ejercicios

1.Escriba su nombre en la pantalla con caracteres intermitentes. Escoja para uno de los colores el mismo del fondo, de manera que el nombre aparezca y desaparezca.

2.Dibuje una llama con líneas de colores apropiados. (Podrá conseguir un dibujo muy efectivo utilizando colores intermitentes.)

3.Dibuje un cangrejo rojo reposando en un playa de arenas amarillas. Elija las tintas y los colores de manera que los colores del dibujo no cambien al cambiar el modo de pantalla.

2. Caracteres y códigos

EL JUEGO DE CARACTERES DEL AMSTRAD

El Amstrad dispone de una considerable variedad de caracteres predefinidos. Además de los caracteres alfabéticos y numéricos, tiene símbolos para representar notas musicales, hombrecillos, etc., como se puede comprobar ejecutando el siguiente programa:

```
10 MODE 1
20 FOR codigo=32 TO 255
30   PRINT CHR$(codigo);
40 NEXT
```



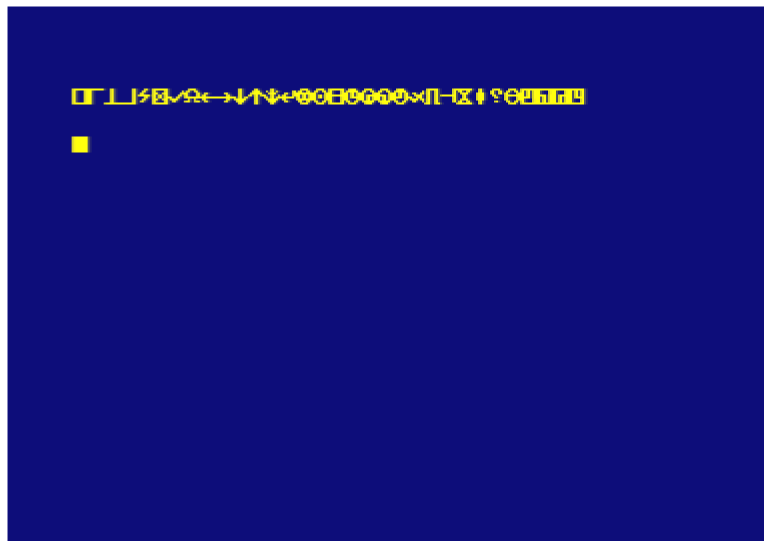
El Amstrad asocia cada uno de estos caracteres con un número de código, llamado "código ASCII" del carácter. Los códigos van del 0 al 255. Los códigos del 0 al 31 tienen significados especiales para el ordenador, tales como 'hacer retroceder un espacio el cursor' o 'cambiar el color'. Los códigos del 32 al 255 corresponden a las letras mayúsculas y minúsculas, los números, los signos de puntuación, etc. La línea 30 de programa hace que el ordenador escriba el carácter que corresponde a un determinado código ASCII. La figura 2.1 muestra la distribución de los códigos ASCII más usuales.

Caracteres	Códigos ASCII
Códigos especiales	0-31
Espacio	32
0-9	48-57
A-Z	65-90
a-z	97-122

Figura 2.1. Algunos de los códigos ASCII más usuales.

Los caracteres con código entre 0 y 31 pueden ser exhibidos en la pantalla precediéndolos del carácter cuyo código ASCII es 1:

```
10 MODE 1
20 FOR codigo=0 TO 31
30   PRINT CHR$(1)CHR$(codigo);
40 NEXT
```



Esto proporciona una notable posibilidad de elección de caracteres para un programa. Sin embargo, hay situaciones en las que se necesita algún carácter que no está incluido en el juego de caracteres predefinidos del Amstrad; así ocurre con las letras de ciertos idiomas, con los símbolos matemáticos y también en los programas de juegos. En estos casos podemos recurrir a la posibilidad que ofrece el Amstrad de crear nuestros propios caracteres definidos por el usuario. Antes de hablar de esto, vamos a echar un vistazo a la forma en que el Amstrad almacena los caracteres y a ver por qué este método de almacenamiento limita los caracteres predefinidos a sólo 256, con códigos ASCII del 0 al 255.

Sistema binario; bits y bytes

¿Cómo almacena el Amstrad la información? Para entenderlo de manera sencilla, lo mejor es imaginarse el ordenador relleno de millares de interruptores, cada uno de los cuales puede estar en las posiciones de 'encendido' y 'apagado'. En el Amstrad, estos "interruptores" están siempre agrupados en bloques de ocho. Representando la posición de 'apagado' por un 0 y la de 'encendido' por un 1, podemos representar las posibles combinaciones de ocho interruptores como se indica en la figura 2.2.

Quizá no le sorprenda ya saber que los ocho interruptores pueden encontrarse en 256 situaciones posibles. Cada uno de los números es un número binario, compuesto únicamente de las cifras 0 y 1. Las cifras 0 y 1 son las cifras o dígitos binarios; cada una de estas cifras es un bit. Una combinación de 8 bits es un byte.

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1
      ...
1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1
```

Figura 2.2. Las 256 combinaciones posibles de 8 "interruptores".

El sistema binario es una forma de contar agrupando en grupos de dos, de la misma manera que nosotros contamos habitualmente haciendo grupos de diez. Cada número binario es equivalente a un número en nuestro sistema de numeración. Es relativamente sencillo convertir un número binario en la forma más familiar de número decimal, como veremos enseguida.

El byte es la unidad fundamental de almacenamiento de información en el Amstrad. Muchas de las limitaciones del ordenador provienen justamente del hecho de que sólo existen 256 bytes diferentes. Si hay sólo 256 caracteres predefinidos es porque cada carácter debe tener asociado un código ASCII que se almacene en un byte. Una línea de programa puede tener una longitud de entre 0 y 255 caracteres, ya que su número de caracteres se almacena en un byte. Si existiesen 257 caracteres predefinidos, o si una línea pudiese tener más de 255 caracteres, estas informaciones deberían almacenarse en dos bytes, lo que requeriría considerable cantidad de memoria suplementaria.

Cada carácter predefinido tiene un código ASCII, pero este código no es suficiente para que el ordenador sepa cómo debe representar el carácter en la pantalla. Cada carácter está construido en una cuadrícula de 8 x 8, como vemos en el ejemplo de la letra "A" mayúscula.

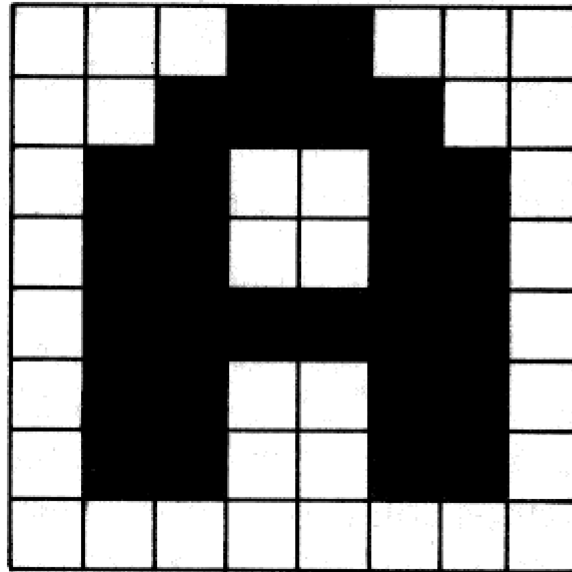


Figura 2.3. La letra "A" mayúscula.

Cada elemento de la cuadrícula puede estar encendido o apagado (en la pantalla) y aquí entra de nuevo el sistema binario, ¿verdad? Por otra parte, la cuadrícula 8 x 8 es una consecuencia de los 8 bits de un byte. Cada fila de la cuadrícula puede ser almacenada en un byte, y la descripción completa del carácter ocupa 8 bytes.

```
0 0 0 1 1 0 0 0
0 0 1 1 1 1 0 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 1 1 1 1 1 1 0
0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0
0 0 0 0 0 0 0 0
```

Figura 2.4. Definición binaria, con 8 bytes, del carácter "A".

Los números binarios son incómodos para trabajar con ellos porque son largos y es fácil añadir y omitir por error algún 0 o algún 1. Es fácil convertir en número decimal el valor binario de un byte; basta con sumar los números que aparecen sobre las posiciones del byte en las que hay un 1.

128	64	32	16	8	4	2	1		
0	0	0	1	1	0	0	0	16+8	= 24
0	0	1	1	1	1	0	0	32+16+8+4	= 60
0	1	1	0	0	1	1	0	64+32+4+2	= 102
0	1	1	0	0	1	1	0	64+32+4+2	= 102
0	1	1	1	1	1	1	1	64+32+16+8+4+2	= 126
0	1	1	0	0	1	1	0	64+32+4+2	= 102
0	1	1	0	0	1	1	0	64+32+4+2	= 102
0	0	0	0	0	0	0	0	0	= 0

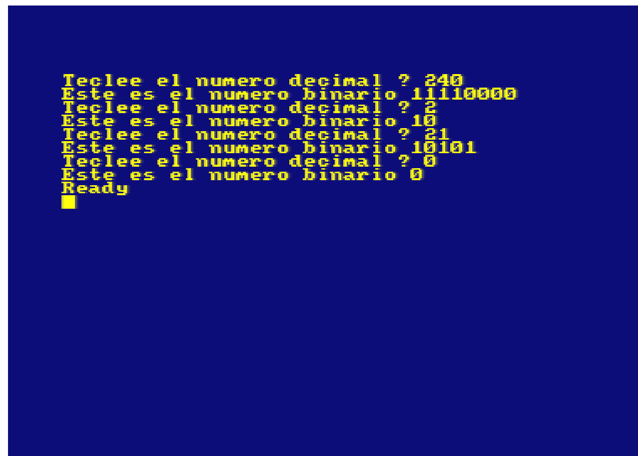
Figura 2.5. Definición, con números decimales, del carácter "A".

El Amstrad puede facilitar aún más este proceso realizando el trabajo más tedioso. El comando PRINT STR$ (número) convierte el número que incluyamos como parámetro en una cadena literal que es equivalente en forma decimal. Podemos comprobar que nuestros cálculos eran correctos mediante el programa siguiente. Hay que poner los símbolos "&X" antes de un número binario; de no hacerlo así, el Amstrad lo tomaría como un número decimal formado por sólo ceros y unos.

```
10 MODE 1
20 numero=1
30 WHILE numero>0
40   INPUT "Teclee el numero binario, precedido por &X ";numero
50   PRINT "Este es el numero decimal "STR$(numero)
60 WEND
```

A veces también es necesario convertir números decimales en binarios. También de eso se puede encargar el Amstrad:

```
10 MODE 1
20 numero=1
30 WHILE numero>0
40   INPUT "Teclee el numero decimal ";numero
50   PRINT "Este es el numero binario "BIN$(numero)
60 WEND
```



Respecto de estos comandos que convierten números, hay que advertir que proporcionan una cadena literal como resultado. No se pueden hacer operaciones aritméticas con cadenas; para realizarlas hay que convertir previamente la cadena literal en un número:

```
10 MODE 1
20 numero=1
30 WHILE numero>0
40   INPUT "Teclee el numero decimal ";numero
50   PRINT "Este es el numero binario "BIN$(numero)
54   REM VAL convierte una cadena literal en un numero
55   numerico=VAL(BIN$(numero))
56   PRINT "Este es el numero ";numerico
60 WEND
```

Cómo definir sus propios caracteres

Cualquiera de los 16 caracteres cuyo código ASCII está entre 240 y 255 se puede redefinir para que tenga otra forma diferente. Para ver cómo hacerlo, vamos a cambiar el carácter 240 para que sea la letra "A", aprovechando el hecho de que ya conocemos los valores de los 8 bytes que se utilizan en la descripción de esta

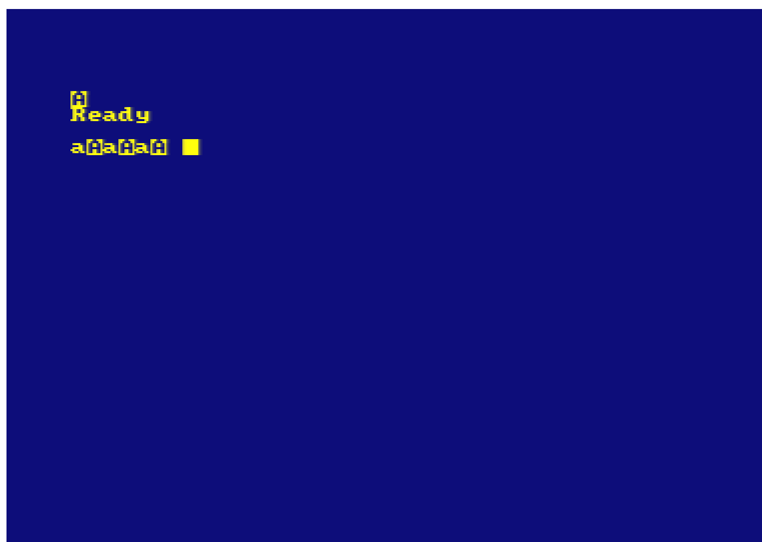
letra:

```
10 MODE 1
20 REM SYMBOL define el caracter
30 SYMBOL 240,24,60,102,102,126,102,102,0
40 PRINT CHR$(240)
```

La instrucción SYMBOL de la línea 30 es la que se encarga de informar al Amstrad de nuestra nueva definición del carácter. El primer número, 240, es el del código ASCII del carácter; los ocho números que siguen definen, fila por fila, la cuadrícula del carácter.

Para redefinir otros códigos ASCII además de los 16 citados, hay que emplear la instrucción SYMBOL AFTER, como en el programa siguiente:

```
10 MODE 1
20 SYMBOL AFTER 65
30 SYMBOL 65,231,195,153,153,129,153,153,255
40 PRINT CHR$(65)
```



La línea 20 indica al ordenador que vamos a redefinir caracteres ASCII con código igual o superior a 65. La 30 redefine el código 65, que inicialmente representa la letra "A", para que dé una imagen en negativo de la letra. El carácter que había antes de la definición ha desaparecido. Para recuperarlo se puede reinicializar el ordenador, o también se puede emplear la instrucción SYMBOL AFTER, que devuelve a todos los caracteres posteriores al especificado su aspecto inicial:

```
10 MODE 1
20 SYMBOL AFTER 65
30 SYMBOL 65,231,195,153,153,129,153,153,255
```

```
40 PRINT CHR$(65)
50 SYMBOL AFTER 70
```

Sistema hexadecimal

Ya hemos visto que podemos utilizar el ordenador para facilitarnos la conversión entre números binarios y decimales. Aunque para nosotros el sistema familiar sea el decimal, cuando se trabaja con ordenadores es usual utilizar el sistema hexadecimal, que es el sistema de base 16.

NUMERO DECIMAL	EQUIVALENTE HEXADECIMAL
0	0
1	1
...	...
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
...	...
30	1E
...	...
100	64
...	...
255	FF

Figura 2.6. Algunos números decimales y sus equivalentes hexadecimales.

Los números hexadecimales se escriben habitualmente precedidos del símbolo '&' para que no se confundan con los decimales. Es conveniente acostumbrarse a trabajar con este sistema de numeración. Su ventaja principal estriba en que los valores que puede tomar un byte corresponden exactamente a los números hexadecimales de 2 cifras. Por ejemplo, la descripción de la letra 'A' es más sencilla cuando la hacemos con números hexadecimales, como vemos en el programa siguiente:

```
10 MODE 1
20 SYMBOL 240,&18,&3C,&66,&66,&7E,&66,&66,0
30 PRINT CHR$(240)
```

Puesto que el ordenador trabaja con bytes, los números hexadecimales resultan más convenientes en la mayor parte de los casos. Así, bastantes de los números que nos aparecen repetidamente, y cuya significación no queda clara al

escribirlos en decimal, revelan su importancia al escribirlos en hexadecimal. Por ejemplo, el máximo valor posible de los números de línea en los programas, 65535, parece arbitrario escrito en decimal. Su versión hexadecimal, &FFFF, aclara la razón de este número límite; 65535 es el mayor número que se puede escribir con dos bytes. Con números de línea almacenados en dos bytes se puede efectivamente llegar hasta 256 x 256 = 65535. El empleo de números más grandes requeriría que se les dedicasen al menos tres bytes. En cambio, si los números de línea se almacenasen en un byte, su valor sólo podría estar entre 0 y 255.

El Amstrad convierte sin dificultad números decimales en hexadecimales:

```
10 MODE 1
20 numero=1
30 WHILE numero>0
40   INPUT "Teclee el numero decimal ";numero
50   PRINT "En hexadecimal vale "HEX$(numero)
60 WEND
```

También hace la conversión inversa; en este caso, el número que se introduce es hexadecimal, y debe ir precedido de '&':

```
10 MODE 1
20 numero=1
30 WHILE numero>0
40   INPUT "Teclee el numero hexadecimal, precedido por & ";Numero
50   PRINT "Este es el numero decimal "STR$(numero
60 WEND
```

JUEGOS Y CARACTERES DEFINIDOS

Los caracteres definidos por el usuario se emplean con gran frecuencia en los programas de juegos. El programa que sigue define un carácter con forma de perro y lo escribe en varios lugares de la pantalla elegidos aleatoriamente:

```
10 MODE 0
20 SYMBOL 240,0,4,7,132,124,130,130,0
30 FOR perroaleatorio=1 TO 30
40   xaleat=INT(19*RND(1)+1)
50   yaleat=INT(24*RND(1)+1)
60   tintapluma=INT(15*RND(1)+1)
70   PEN tintapluma
80   LOCATE xaleat,yaleat
90   PRINT CHR$(240)
100 NEXT
```



Para hacer que la figura se mueva por la pantalla, vamos a escribir un espacio en blanco en la posición que ocupa el perro (para borrarlo) y a escribir el perro en una nueva posición. El movimiento se dirige mediante las teclas 'a', 'z', ',' y '.' y el programa se termina cuando se pulsa 'e'.

```
10 MODE 0
20 SYMBOL 240,0,4,7,132,124,130,130,0
30 perro$=CHR$(240)
40 PEN 1
50 xperro=13:yperro=10
60 respuesta$=""
70 WHILE respuesta$<>"e"
80   ynuevo=yperro:xnuevo=xperro
90   respuesta$=INKEY$
100   IF respuesta$="a" AND yperro>1 THEN ynuevo=yperro-1
110   IF respuesta$="z" AND yperro<25 THEN ynuevo=yperro+1
120   IF respuesta$="," AND xperro>1 THEN xnuevo=xperro-1
130   IF respuesta$="." AND xperro<20 THEN xnuevo=xperro+1
140   IF xperro<>xnuevo OR yperro<>ynuevo THEN LOCATE xperro,yperro:PRINT " ";:xperro=xnuevo:yperro=ynuevo
150   LOCATE xperro,yperro
160   PRINT perro$
170 WEND
```

El movimiento resulta más creíble cuando se utilizan caracteres distintos para las diferentes direcciones del movimiento. Para demostrarlo podríamos modificar el programa anterior, pero lo que haremos será utilizar como caracteres los que ya están definidos para los códigos ASCII 240 a 243. Se trata de flechas que apuntan en las cuatro direcciones:

```
10 MODE 0
30 flecha$=CHR$(240)
40 PEN 1'
50 xflecha=13:yflecha=10
60 respuesta$=""
70 WHILE respuesta$<>"e"
80   ynuevo=yflecha:xnuevo=xflecha
90   respuesta$=INKEY$
100   IF respuesta$="a" AND yflecha>1 THEN ynuevo=yflecha-1:flecha$=CHR$(240)
110   IF respuesta$="z" AND yflecha<25 THEN ynuevo=yflecha+1:flecha$=CHR$(241)
120   IF respuesta$="," AND xflecha>1 THEN xnuevo=xflecha-1:flecha$=CHR$(242)
130   IF respuesta$="." AND xflecha<20 THEN xnuevo=xflecha+1:flecha$=CHR$(243)
140   IF xflecha<>xnuevo OR yflecha<>ynuevo THEN LOCATE xflecha,yflecha:PRINT "
";:xflecha=xnuevo: yflecha=ynuevo
150   LOCATE xflecha,yflecha
160   PRINT flecha$
170 WEND
```

La idea de este programa puede servir de base para muchos juegos.

Uno de los problemas que se presentan al definir caracteres es que, cuando se dibuja en papel un esquema del carácter, el posterior resultado en pantalla resulta con frecuencia bastante diferente de lo esperado. El diseñador de caracteres que figura en la cinta de presentación del Amstrad tiene el grave inconveniente de no dar la definición numérica del carácter que se dibuja; esta definición es necesaria para la sentencia SYMBOL.

El programa que sigue le permitirá crear sus propios caracteres, le proporcionará la definición numérica del carácter creado y le dará la oportunidad de grabar esta definición en un fichero. Así podrá construir su propia biblioteca de caracteres definidos, para utilizarla en la elaboración de programas.

```
10 MODE 1
20 DEFINT c,n,x,y
29 REM matrices para almacenar la definicion del caracter
30 DIM codigo(8,8),simbolo(8)
40 INK 3,20,6
50 nombre$= "??????"
60 numero=240
69 REM
70 GOSUB 1000
80 GOSUB 3000:GOSUB 4000
89 REM esperar la respuesta del teclado
90 respuesta$=""
100 WHILE respuesta$<>"e"
110 xnuevo=x:ynuevo=y
120 respuesta$=LOWER$(INKEY$)
129 REM las cuatro lineas siguientes controlan e 1 movimiento del cursor arriba/
```

```
abajo/izquierda/d erecha
130 IF respuesta$="a" AND y>ycomienzo THEN ynuevo=y-1
140 IF respuesta$="z" AND y<ycomienzo+7 THEN ynuevo=y+1
150 IF respuesta$="," AND x>xcomienzo THEN xnuevo=x-1
160 IF respuesta$="." AND x<xcomienzo+7 THEN xnuevo=x+1
169 REM se rectifica la posicion si ha cambiado
170 IF xnuevo<>x OR ynuevo<>y THEN GOSUB 2000
179 REM si se aprieta la barra de espacio cambia el color del punto
180 IF respuesta$=" " THEN GOSUB 7000:GOSUB 3000
189 REM si se pulsa 'o' se graba la definicion del simbolo en un fichero
190 IF respuesta$="o" THEN GOSUB 5000:GOSUB 1000:GOSUB 3000:GOSUB 4000
199 REM si se pulsa 'i' se carga la definicion del simbolo desde la cinta
200 IF respuesta$="i" THEN GOSUB 6000:GOSUB 3000:GOSUB 4000
210 PEN 14
220 LOCATE x,y
230 PRINT CHR$(203);
240 WEND
250 END
999 REM definicion del simbolo vacio
1000 CLS
1010 SYMBOL numero,0,0,0,0,0,0,0,0
1020 PEN 1
1030 FOR con=1 TO 8
1040   FOR con1=1 TO 8
1050     codigo(con,con1)=1
1060   NEXT
1070 NEXT
1079 REM impresion de 8*8 cuadrados vados para representar el caracter 'espacio
en blanco'
1080 xcomienzo=2:ycomienzo=2
1090 FOR x=xcomienzo TO xcomienzo+7
1100   FOR y=ycomienzo TO ycomienzo+7
1110     LOCATE x,y
1120     PRINT CHR$(233);
1130   NEXT
1140 NEXT
1150 x=xcomienzo:y=ycomienzo
1160 RETURN
1999 REM impresion del caracter con su color correcto
2000 LOCATE x,y
2010 xcodigo=x-xcomienzo+1:ycodigo=y-ycomienzo+1
2020 PEN codigo (xcodigo,ycodigo)
2030 PRINT CHR$(233);
2040 x=xnuevo:y=ynuevo
2050 RETURN
2999 REM conversion de la matriz codigo en los o cho numeros decimales que
definen el caracter
```

```
3000 FOR con=1 TO 8
3010   simbolo$="&X"
3020   FOR con1=1 TO 8
3030     simbolo$=simbolo$+MID$(STR$(codigo(con1,con )-1),2,1)
3040   NEXT
3050   simbolo(con)=VAL(simbolo$)
3060 NEXT
3070 SYMBOL
numero,simbolo(1),simbolo(2),simbolo(3),simbolo(4),simbolo(5),simbolo(6),simbolo
(7), simbolo(8)
3080 PEN 1
3090 LOCATE 1,ycomienzo+10
3100 PRINT "Simbolo: ";CHR$(numero);
3110 LOCATE 1,ycomienzo+12
3120 PRINT "SYMBOL
";numero;simbolo(1);simbolo(2);simbolo(3);simbolo(4);simbol0(5);simbolo(6);simbo
lo(7);simbolo(8);
3130 RETURN
4000 PEN 1
4010 LOCATE 1,ycomienzo+15
4020 PRINT "Nombre del simbolo: ";nombre$
4030 LOCATE 1,ycomienzo+17
4040 PRINT "Simbolo numero: ";numero
4050 RETURN
4999 REM grabar la definicion en un fichero 5000 LOCATE 1,21
5010 PEN 1,
5020 INPUT "Nombre del simbolo";nombre$
5030 OPENOUT nombre$
5040 WRITE
#9,nombre$,numero,simbolo(1),simbolo(2),simbolo(3),simbolo(4),simbolo(5),simbolo
(6),simbolo(7),simbolo(8)
5050 CLOSEOUT
5059 REM nombre y numero de simbolo para la sigu lente definicion
5060 nombre$="??????"
5070 numero=numero+1
5080 RETURN
CARACTERES Y CÓDIGOS 31
5999 REM lectura del fichero con la definicion d el caracter
6000 LOCATE 1,21
6010 PEN 1
6020 INPUT "Nombre del simbolo";nombre$
6030 OPENIN nombre$
6040 INPUT #9,nombre$,numero,simbolo(1),simbolo(
2),simbolo(3),simbolo(4),simbolo(5),simbolo(6),simbolo(7),simbolo(8)
6050 CLOSEIN
6060 CLS
6069 REM conversion a binario de la matriz simbolo para obtener la matriz codigo
```

```
6070 FOR con=1 TO 8
6080   simbolo$=BIN$(simbolo(con))
6090   longitud=LEN(simbolo$)
6100   simbolo$=STRING$(8-longitud,"0")+simbolo$
6110   FOR con1=1 TO 8
6120     LOCATE xcomienzo+con1-1,ycomienzo+con-1
6130     codigo=VAL(MID$(simbolo$,con1,1))+1
6140     PEN codigo
6150     PRINT CHR$(233);
6160     codigo(con1,con)=codigo
6170   NEXT
6180 NEXT
6190 RETURN
6999 REM si se aprieta la barra de espacio cambia el color en la posicion del cursor
7000 xcodigo=x-xcomienzo+1:ycodigo=y-ycomienzo+1
7010 IF codigo(xcodigo,ycodigo)=1 THEN codigo(xcodigo,ycodigo)=2 ELSE codigo(xcodigo,ycodigo)=1
7020 RETURN
```



Ejercicios

1.Diseñe un carácter con forma de araña y defina la tecla de '.' para que proporcione arañas al pulsarla.

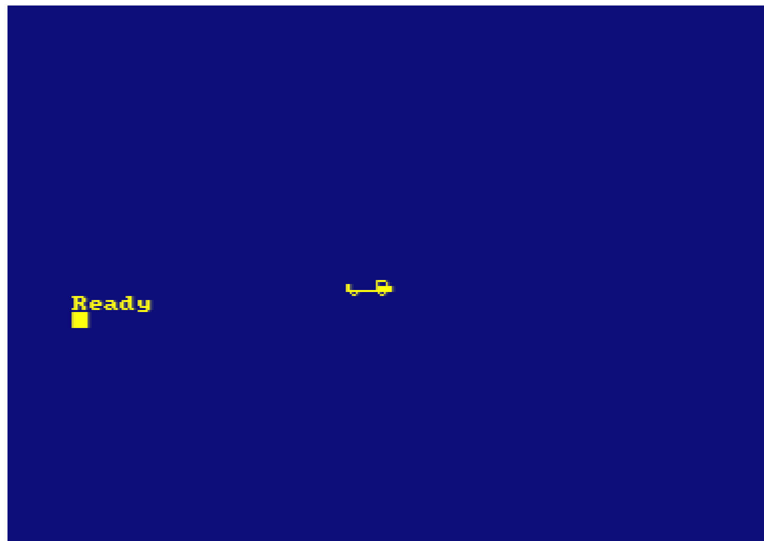
2.Escriba un programa que mueva su araña por la pantalla. A base de líneas podrá además dibujar una telaraña.

3.Mejore ahora su programa diseñando arañas que miren en las cuatro direcciones y dibujando la que convenga para que la araña vaya mirando en la dirección del movimiento.

Figuras de varios caracteres

En el modo 1, el tamaño de un carácter no da mucho de sí para crear una figura detallada; puede ser conveniente construir una figura mayor yuxtaponiendo varios caracteres. Cuando la figura es horizontal, se pueden juntar varios caracteres en una cadena literal:

```
10 MODE 1
19 REM los tres caracteres que forman el camion
20 SYMBOL 240,0,0,96,96,96,127,18,12
30 SYMBOL 241,0,0,0,0,0,255,0,0
40 SYMBOL 242,248,132,132,255,255,255,72,48
50 camion$=CHR$(240)+CHR$(241)+CHR$(242)
60 LOCATE 18,13
70 PRINT camion$
```



Es fácil mover el camión con ayuda del teclado, aunque para que resulte convincente sólo lo moveremos hacia la derecha; para ello hay que pulsar

```
10 MODE 1
20 REM los tres caracteres que forman el camion
30 SYMBOL 240,0,0,96,96,96,127,18,12
40 SYMBOL 241,0,0,0,0,0,255,0,0
50 SYMBOL 242,248,132,132,255,255,255,72,48
60 camion$=CHR$(240)+CHR$(241)+CHR$(242)
70 x=1:y=13
80 LOCATE x,y
90 PRINT camion$
100 respuesta$=""
109 REM
110 WHILE respuesta$<>"e"
120 xnuevo=x
```

```
130 respuesta$=LOWER$(INKEY$)
140 IF respuesta$="." THEN xnuevo=x+1
149 REM si el camion se ha movido se borra con un espacio el antiguo caracter de la izquierda
150 IF xnuevo<>x THEN LOCATE x,y:PRINT " ";camion$
160 x=xnuevo
170 WEND
```

Observe lo que ocurre cuando la figura llega al final de la línea: salta automáticamente al comienzo de la nueva línea. Esto se debe a que el Amstrad nunca escribe un carácter en una posición que le lleve fuera de la pantalla. Cuando se le presenta esta situación, el ordenador toma la decisión de mover el cursor de texto a una posición conveniente, de acuerdo con las siguientes reglas:

1.Si el cursor se va a salir de la pantalla por la derecha, se lo traslada al comienzo de la línea siguiente.

2.Si el cursor se va a salir de la pantalla por la izquierda, se lo traslada a la última posición de la línea anterior.

3.Si el cursor se va a salir por la parte superior de la pantalla, el contenido de ésta se desplaza una línea hacia abajo y el cursor permanece en la primera línea de la pantalla.

4.Si el cursor se va a salir por la parte inferior de la pantalla, el contenido de ésta se desplaza una línea hacia arriba, y el cursor permanece en la última línea de la pantalla.

Aunque sólo el carácter delantero del camión se sitúe en una posición ilegal, el ordenador considera toda la cadena literal como situada ilegalmente. Esto ocurre cuando el camión comienza en la posición 39 de la línea; el ordenador lleva entonces toda la cadena al comienzo de la línea siguiente. Es importante recordar esto, ya que significa que el tamaño de un carácter múltiple influye decisivamente en el número de posiciones en las que podrá ser dibujado. En el caso que nos ocupa, el mayor valor que puede tomar la coordenada x de texto es 38.

Parece bastante más difícil mover una figura compuesta cuando tiene forma vertical, puesto que en principio habría que emplear una instrucción LOCATE para situar cada uno de los caracteres que la componen. Sin embargo, viene en nuestra ayuda el hecho de que existen cuatro códigos ASCII (de valores inferiores a 32) que sirven para mover el cursor en las cuatro direcciones. Son los que se explican en la figura 2.7. Incluyendo estos códigos como caracteres, es posible describir una figura vertical en forma de cadena literal; es lo que hacemos en la figura 2.8. El ordenador no dibujará en la pantalla estos caracteres de movimiento del cursor, sino que realizará con el cursor de texto el movimiento que corresponda.

Código ASCII	Acción
8 9 10 11	Mover el cursor una posición hacia atrás Mover el cursor una posición hacia delante Mover el cursor una línea hacia abajo Mover el cursor una línea hacia arriba

Figura 2.7. Los cuatro códigos ASCII de movimiento del cursor.

El cohete se puede representar mediante una cadena literal compuesta por:

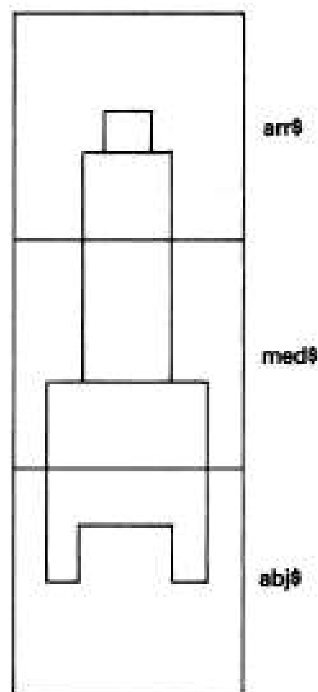
```
arr$ + CHR$(8) + CHR$(10)
```

(mueve el cursor atrás y hacia abajo para escribir los siguientes caracteres)

```
+med$ + CHR$(8) + CHR$(10)
```

(lo mismo otra vez)

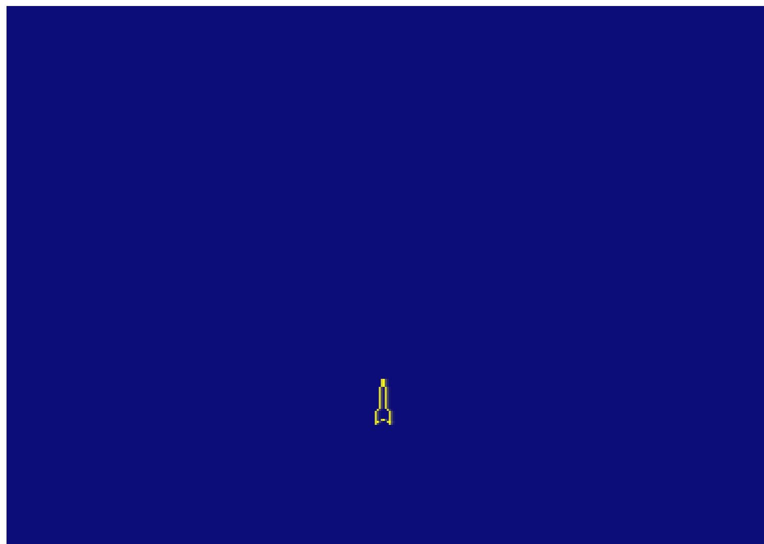
```
+abj$
```



Para colocar el cohete en la posición deseada de pantalla, basta con una sola instrucción LOCATE:

```
10 MODE 1
19 REM los tres caracteres que forman el cohete
```

```
20 SYMBOL 240,0,24,24,24,24,36,36,36
30 SYMBOL 241,36,36,36,36,36,36,36,36
40 SYMBOL 242,66,129,129,129,129,153,195,129
50 cohete$=CHR$(240)+CHR$(8)+CHR$(10)+CHR$(241)+ CHR$(8)+CHR$(10)+CHR$(242)
60 x=20:y=20
70 LOCATE x,y
80 PRINT cohete$
90 respuesta$=""
100 REM leer del teclado en tanto no se pulse 'e
110 WHILE respuesta$<>"e" AND y>1
120 ynuevo=y
130 respuesta$=LOWER$(INKEY$)
140 IF respuesta$="a" THEN ynuevo=y-1
150 REM si se ha movido el cohete se borra con un blanco el antiguo caracter
inferior
160 IF ynuevo<>y THEN LOCATE x,y+2:PRINT " ":LOCATE x,ynuevo:PRINT cohete$
170 y=ynuevo
180 WEND
```



Lamentablemente, cuando el Amstrad echa sus cuentas sobre la longitud de una cadena y lo que le resta de línea, no tiene en cuenta los posibles caracteres de control del cursor que pueda haber en la cadena. En el caso anterior, el ordenador considerará que debe escribir una cadena con 7 caracteres y, en consecuencia, no podrá hacerlo en una posición cuya coordenada X de texto sea superior a 34. Puede comprobar lo que decimos con sólo cambiar la línea 60:

```
60 x=35:y=20
```

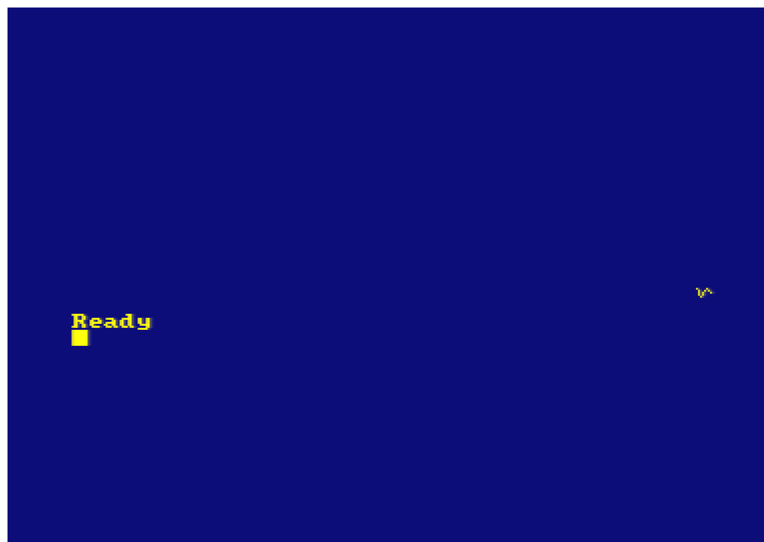
Así pues, podemos usar los caracteres de control del cursor para crear figuras múltiples, pero teniendo siempre en cuenta lo que acabamos de decir. En muchas

ocasiones será preferible utilizar varias sentencias LOCATE para evitarse estos problemas. El siguiente programa muestra conjuntamente las dos posibilidades:

```
10 MODE 1
19 REM los tres caracteres que definen el cohete
20 SYMBOL 240,0,24,24,24,24,36,36,36
30 SYMBOL 241,36,36,36,36,36,36,36,36
40 SYMBOL 242,66,129,129,129,129,153,195,129
49 REM definicion del cohete usando movimientos del cursor
50 cohete$=CHR$(240)+CHR$(8)+CHR$(10)+CHR$(241)+ CHR$(8)+CHR$(10)+CHR$(242)
59 REM con este procedimiento la coordenada x no debe pasar de 34
60 x=1:y=20
70 LOCATE 3,23
80 PRINT "Definido con movimientos del cursor"
90    FOR ycoord=y TO 1 STEP -1
100   LOCATE x,ycoord+3
110   PRINT " "
120   LOCATE x,ycoord
130   PRINT cohete$
140 NEXT
149 REM hay que pulsar una tecla para continuar la demostracion
150 respuesta$=""
160 WHILE respuesta$=""
170 respuesta$=LOWER$(INKEY$)
180 WEND
190 CLS
199 REM con este procedimiento la coordenada s p uede tomar cualquier valor
200 x=36:y=20
210 LOCATE 7,23
220 PRINT "Definido usando LOCATE para cada parte"
229 REM cada parte del cohete se coloca mediante su propia instruccion LOCATE
230 FOR ycoord=y TO 1 STEP -1
240   LOCATE x,ycoord
250   PRINT CHR$(240)
260   LOCATE x,ycoord+1
270   PRINT CHR$(241)
280   LOCATE x,ycoord+2
290   PRINT CHR$(242)
299   REM Se horra la base del cohete de la posicion anterior
300   LOCATE x,ycoord+3
310   PRINT " "
320 NEXT
```

Se pueden obtener efectos más interesantes definiendo dos figuras ligeramente diferentes y dibujando alternativamente una u otra. Utilizando esta idea, producimos el efecto de una culebra que serpentea sobre la pantalla:

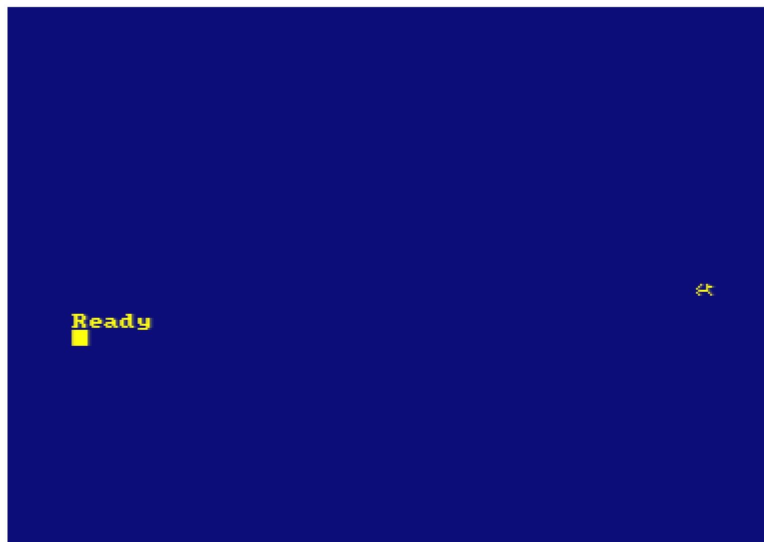
```
10 MODE 1
19 REM definicion de los dos caracteres con forma de culebra
20 SYMBOL 240,0,0,32,80,81,74,130
30 SYMBOL 241,0,0,0,132,74,81,80,32
40 culebra1$=CHR$(240)
50 culebra2$=CHR$(241)
60 culebra$=culebra1$
70 y=13
80 FOR x=1 TO 39
89    REM alternancia entre los dos caracteres culebra
90    IF culebra$=culebra1$ THEN culebra$=culebra2$ ELSE culebra$=culebra1$
100   LOCATE x,y
109   REM el espacio borra la culebra anterior
110   PRINT " ";culebra$
119   REM un tiempo de espera para que el movimiento no sea excesivamente rapido
120   tiempo=TIME
130   WHILE TIME<tiempo+30
140   WEND
150 NEXT
```



De la misma manera podríamos definir dos figuras de camión para dar la impresión del traqueteo que produce su movimiento. O podemos hacer que el perro que ya habíamos dibujado menee ahora la cola al pasearse:

```
10 MODE 1
19 REM definicion de los dos caracteres con form a de perro
20 SYMBOL 240,0,132,135,132,124,130,130,0
30 SYMBOL 241,0,36,71,132,124,130,65,0
40 perro1$=CHR$(240)
```

```
50 perro2$=CHR$(241)
60 perro$=perro1$
70 y=13
80 FOR x=1 TO 39
89    REM alternancia entre los dos caracteres perr o
90    IF perro$=perro1$ THEN perro$=perro2$ ELSE perro$=perro1$
100   LOCATE x,y
109   REM el espacio borra el perro anterior
110   PRINT " ";perro$
119   REM un tiempo de espera para que el movimiento no sea excesivamente rapido
120   tiempo=TIME
130   WHILE TIME<tiempo+30
140   WEND
150 NEXT
```



Ejercicios

1.Dibuje su propia versión de un autobús con varios caracteres y condúzcalo a lo largo de la pantalla.

2.Diseñe dos caracteres circulares con un diámetro situado en dos ángulos diferentes y dibújelos alternativamente para obtener la impresión de una rueda.

3.Añada a su autobús unas ruedas que giren al desplazarlo.

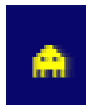
Cómo mejorar la resolución

A pesar de que se pueden lograr muchas cosas utilizando únicamente la pantalla de texto, hay ciertas limitaciones que resultan obvias. La mejor resolución posible es la del modo 2, pero está limitada a 25 líneas de 80 caracteres. En cambio, la peor de las resoluciones gráficas es de 160 por 200 puntos.

Afortunadamente, el Amstrad permite dibujar los caracteres en cualquiera de las posiciones de la pantalla gráfica; esto permite realizar dibujos que se muevan con suavidad. También hace posible, en trabajos más profesionales, colocar rótulos en las gráficas en la posición más conveniente, en lugar de tener que hacerlo en la posición de texto más cercana.

El cambio de texto a coordenadas gráficas trae consigo algunos cambios. Cuando se ha ejecutado el comando TAG (de "Text At Graphics", texto en posiciones gráficas), los caracteres no se colocan en su sitio mediante la sentencia LOCATE, sino mediante MOVE, que sirve para situar el cursor gráfico. El punto que ocupa el cursor gráfico será el punto superior izquierdo del carácter. Veamos cómo podemos mover de esta forma un carácter que representa uno de los clásicos "invasores del espacio":

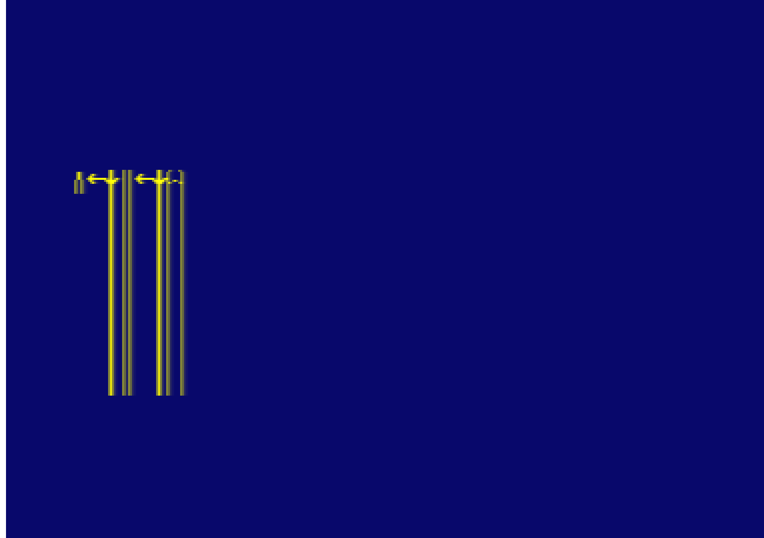
```
10 MODE 1
19 REM definicion del invasor extraterrestre
20 SYMBOL 240,24,60,126,219,255,255,165,165
30 invasor$=CHR$(240)
40 xgrafico=100:ygrafico=200
49 REM asociar el texto al ·cursor grafito
50 TAG
60 FOR x=xgrafico TO 600
70   MOVE x,ygrafico
78   REM el espacio en blanco es para borrar el invasor de la posicion anterior
79   REM elimine el punto y coma para probar
80   PRINT " "invasor$;
90 NEXT
```



Es fundamental aquí el punto y coma que termina la sentencia PRINT, ya que, cuando se utiliza TAG para controlar la escritura de caracteres, los códigos entre 0 y 31 no son obedecidos, sino dibujados. Sin el punto y coma, se dibujarían en cada caso los caracteres 10 y 13 que, como caracteres de control, son CR y LF (los que sirven para llevar el cursor al comienzo de la línea siguiente). Lo verá aún mejor con el ejemplo del cohete:

```
10 MODE 1
19 REM los tres caracteres que definen el cohete
20 SYMBOL 240,0,24,24,24,24,36,36,36
30 SYMBOL 241,36,36,36,36,36,36,36,36
40 SYMBOL 242,66,129,129,129,129,153,195,129
50 cohete$=CHR$(240)+CHR$(8)+CHR$(10)+CHR$(241)+CHR$(8)+CHR$(10)+CHR$(242)
60 TAG
70 xgrafico=300:ygrafico=100
```

```
80 FOR y=ygrafico TO 350
89    REM se borra la base del antiguo cohete
90    MOVE xgrafico,y-24
100   PRINT " ";
109   REM imprimir el nuevo cohete(que birria)
110   MOVE xgrafico,y
120   PRINT cohete$;
130 NEXT
```



¡Nada que ver con lo que queríamos! Los problemas que crea el movimiento del cursor, tanto en el modo de texto como en el gráfico, son más sencillos de resolver para figuras sencillas. Si una figura está formada por una tira horizontal de caracteres, entonces se la puede situar con una sola instrucción MOVE, ya que, al escribir un carácter, el cursor gráfico se mueve automáticamente el espacio equivalente a la anchura de un carácter, colocándose en la posición adecuada para el siguiente. Nuestro camión funcionará sin problemas por este método:

```
10 MODE 1
19 REM los tres caracteres que forman el camion
20 SYMBOL 240,0,0,96,96,96,127,18,12
21 SYMBOL 241,0,0,0,0,0,255,0,0
22 SYMBOL 242,248,132,132,255,255,255,72,48
30 camion$=CHR$(240)+CHR$(241)+CHR$(242)
40 xgrafico=0:ygrafico=200
49 REM asociar el texto al cursor grafico
50 TAG
60 FOR x=xgrafico TO 600
70   MOVE x,ygrafico
79   REM el espacio en blanco es para borrar la parte trasera del camion
anterior
80   PRINT " "camion$;
```

```
90 NEXT
```

Para el cohete, sin embargo, hay que utilizar una serie de instrucciones MOVE:

```
10 MODE 1
19 REM los tres caracteres que definen el cohete
20 SYMBOL 240,0,24,24,24,24,36,36,36
30 SYMBOL 241,36,36,36,36,36,36,36,36
40 SYMBOL 242,66,129,129,129,129,153,195,129
50 cohetearriba$=CHR$(240)
51 cohetemedio$=CHR$(241)
52 coheteabajo$=CHR$(242)
60 TAG
70 xgrafico=300:ygrafico=100
80 FOR y=ygrafico TO 350
89    REM se borra la base del antigua cohete
90    MOVE xgrafico,y-48
100   PRINT " ";
110   MOVE xgrafico,y
120   PRINT cohetearriba$;
121   MOVE xgrafico,y-16
122   PRINT cohetemedio$;
123   MOVE xgrafico,y-32
124   PRINT coheteabajo$;
130 NEXT
```

La sentencia TAG se anula mediante la TAGOFF; también se anula automáticamente al finalizar un programa.

Movimientos más rápidos

Ya se habrá dado cuenta del precio que hay que pagar a cambio de obtener movimientos suaves de las figuras: el programa va más lento. Hay una serie de medidas que podemos tomar para que el programa vaya lo más rápidamente posible.

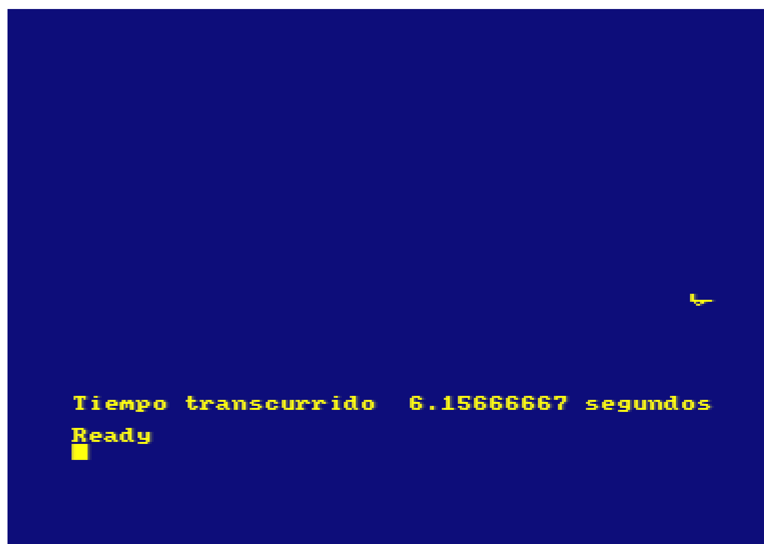
En primer lugar, se aumenta la rapidez si se trabaja con números enteros siempre que sea posible. Esto permite al ordenador hacer los cálculos con mayor rapidez. Puede que piense que no hemos utilizado otra cosa que enteros en los programas precedentes, pero no es así. El ordenador trata todos los números como reales (números con punto decimal) si no se le dice otra cosa. Para declarar ciertas variables como enteras se puede utilizar la instrucción DEFINT:

```
1 DEFINT x,y
```

El ordenador tratará entonces como enteras a todas las variables numéricas que comiencen por G, X o Y. Se puede observar el cambio que esto origina en la

velocidad si se realiza el mismo programa con y sin la especificación de números enteros:

```
1 DEFINT x,y
10 MODE 1
19 REM los tres caracteres que forman el camion
20 SYMBOL 240,0,0,96,96,96,127,19,12
21 SYMBOL 241,0,0,0,0,0,255,0,0
22 SYMBOL 242,240,132,132,255,255,255,72,48
30 camion$=CHR$(240)+CHR$(241)+CHR$(242)
40 xgrafico=0:ygrafico=200
49 REM asociar el texto al cursor grafico
50 TAG
55 tiempoinicial=TIME
60 FOR x=xgrafico TO 600
70   MOVE x,ygrafico
79   REM el espacio en blanca es para borrar la parte trasera del camion
anterior
80   PRINT " "camion$;
90 NEXT
99 tiempototal=TIME
100 TAGOFF
110 LOCATE 1,20
120 PRINT "Tiempo transcurrido "(tiempototal-tiempoinicial)/300 "segundos"
```



Una segunda manera de lograr más rapidez es observar cuál es el mínimo desplazamiento que puede efectuar realmente el ordenador en cada modo. Si en modo 0 se desplaza horizontalmente un carácter en una sola unidad, se vuelve a obtener el mismo carácter. Para obtener un desplazamiento horizontal hay que modificar la coordenada horizontal en al menos 4 unidades en modo O, y en 2 unidades en modo 1. En vertical, la modificación debe ser de 2 unidades en

cualquiera de los modos.

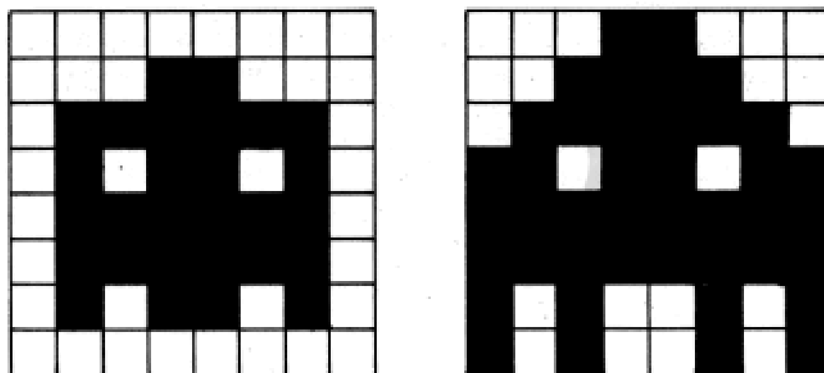


Figura 2.9. Dos caracteres que representan "invasores del espacio"; el primero es más útil, ya que su borde sirve para borrar lo que queda de él en la posición anterior.

Finalmente, podemos diseñar caracteres que tengan un borde vacío alrededor de la figura. Esto garantiza que al desplazarse en cualquier dirección no va dejando huellas tras de sí. La segunda de las dos figuras de "invasores del espacio" que están dibujadas en la figura 2.9 debe ser borrada cada vez que se produce un desplazamiento. Esto contribuye a hacer más lento el programa.

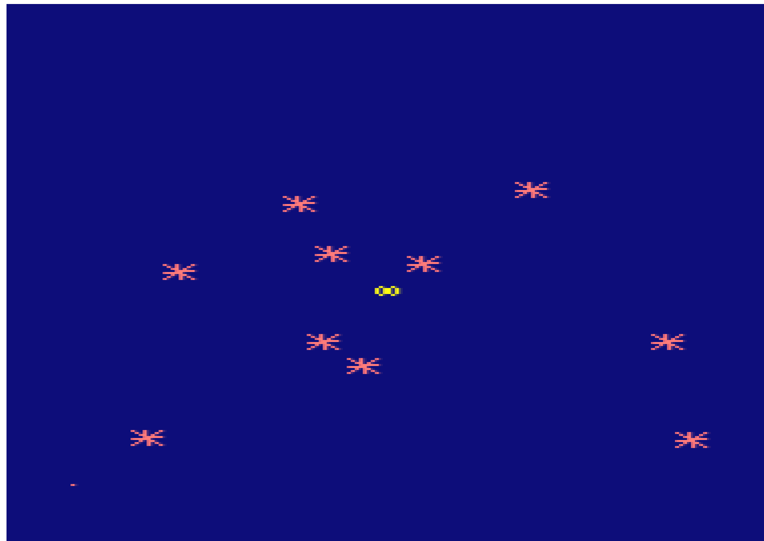
¿Y ahora qué más?

Muchos programas de juegos necesitan identificar lo que hay en una determinada posición de la pantalla. Por ejemplo, ¿cómo saber si un coche choca al correr contra la barrera?; o ¿cómo saber si nuestro disparo ha alcanzado a un invasor?. Utilizando TEST (x,y) para determinar el número de tinta que cubre una determinada posición gráfica. Eligiendo cuidadosamente los colores que se emplean, esto puede servir para que el programa sepa qué es lo que hay en una posición que se ha alcanzado.

Podemos dibujar en rosa todas las arañas, por ejemplo, y hacer que el programa reaccione cuando llevamos nuestra mosca a una posición pintada de rosa:

```
1 REM para aumentar la velocidad
5 DEFINT c,x,y
10 MODE 0
19 REM caracteres mosca y arana
20 SYMBOL 240,0,36,90,90,90,36,0,0
30 SYMBOL 241,145,82,52,31,248,44,74,137
40 mosca$=CHR$(240)
50 arana$=CHR$(241)
60 GOSUB 1000
70 xmosca=300:ymosca=200
80 xnuevo=xmosca:ynuevo=ymosca
```

```
90 xtest=xmosca:ytest=ymosca
100 MOVE mosca,ymosca:PLOT xmosca,ymosca,1
110 GOSUB 2000
120 respuesta$="":moscamuerta=0
129 REM se leen caracteres del teclado mientras la mosca siga viva
130 WHILE respuesta$="" OR moscamuerta=0
140   xnuevo=xmosca:ynuevo=ymosca
150   respuesta$=LOWER$(INKEY$)
159   REM la posicion (xtest,ytest) para comprobacion del color depende de la
direccion del movimiento
160   IF respuesta$="a" THEN ynuevo=ymosca+2:xtest=xnuevo+16:ytest=ynuevo+8
170   IF respuesta$="z" THEN ynuevo=ymosca-2:xtest=xnuevo+16:ytest=ynuevo-24
180   IF respuesta$="." THEN xnuevo=xmosca+4:xtest=xnuevo+48:ytest=ynuevo-8
190   IF respuesta$="," THEN xnuevo=xmosca-4:xtest=xnuevo-16:ytest=ynuevo-8
200   IF xnuevo<>xmosca OR ynuevo<>ymosca THEN GOSUB 2000
210 WEND
220 MODE 1
230 END
999 REM color rosa
1000 MOVE 0,0
1010 DRAW 0,0,11
1020 TAG
1029 REM dibujar 10 aranas aleatoriamente
1030 FOR aranas=1 TO 10
1040   xarana=INT(600*RND(1)+20)
1050   yarana=INT(300*RND(1)+20)
1060   MOVE xarana,yarana
1070   PRINT arana$;
1080 NEXT
1090 RETURN
1999 REM se comprueba el color de la siguiente p osicion del caracter
2000 color=TEST(xtest,ytest)
2008 REM si es rosa, muere la mosca
2009 REM no se considera tocar la arana el tocar solo la suprficie
2010 IF color=11 THEN moscamuerta=1:SOUND 7,2000 2019 REM dibujo de la mosca en
la nueva posicion
2020 xmosca=xnuevo:ymosca=ynuevo
2050 MOVE xmosca,ymosca
2060 PRINT mosca$;
2070 RETURN
```



Podemos modificar el juego utilizando contadores de tiempo:

```
120 respuesta$="":moscamuerta=0
126 REM ahora hay que alcanzar con la mosca. la e squina superior izquierda
127 REM lo mas rapidamente posible, puesto que c uenta el tiempo,
128 tiempoinicial=TIME
129 REM se leen caracteres del teclado mientras la mosca siga viva y no se
alcance la esquina
130 WHILE (respuesta$="" OR moscamuerta=0) AND (xmosca<600 OR ymosca<300)
140   xnuevo=xmosca:ynuevo=ymosca
150   respuesta$=LOWER$(INKEY$)
159   REM la posicion (xtest,ytest) para comprobacion del color depende de la
direccion del movimiento
160   IF respuesta$="a" THEN ynuevo=ymosca+2:xtest=xnuevo+16:ytest=ynuevo+8
170   IF respuesta$="z" THEN ynuevo=ymosca-2:xtest=xnuevo+16:ytest=ynuevo-24
180   IF respuesta$="." THEN xnuevo=xmosca+4:xtest=xnuevo+48:ytest=ynuevo-8
190   IF respuesta$="," THEN xnuevo=xmosca-4:xtest=xnuevo-16:ytest=ynuevo-8
200   IF xnuevo<>xmosca OR ynuevo<>ymosca THEN GOSUB 2000
210 WEND
215 TAGOFF:CLS
220 IF moscamuerta=0 THEN PRINT "Tiempo: "TIME-tiempoinicial
230 END
```

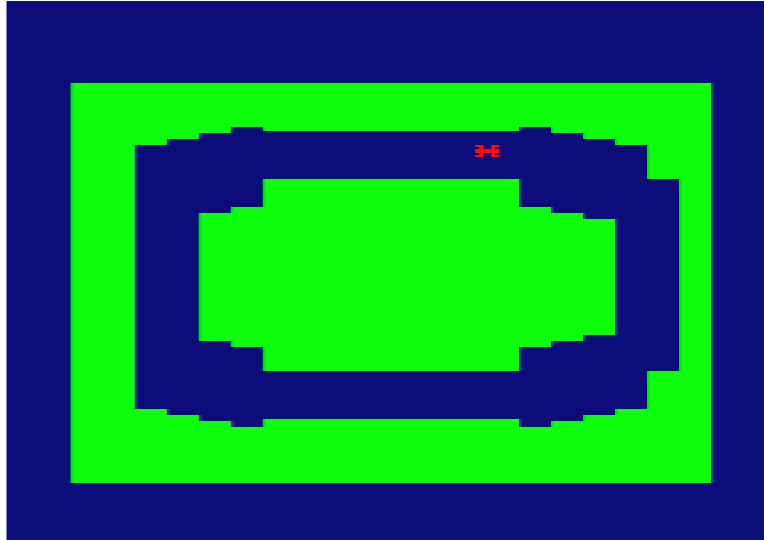
Una variante del comando TEST es el TESTR; la diferencia estriba en que en TESTR la posición que examina se define por su desplazamiento relativo desde la posición actual. Por ejemplo, TESTR (10, -5) examina el punto que se encuentra 10 unidades a la derecha y 5 unidades arriba del actual.

El siguiente programa nos enseña cómo se utiliza; consiste en conducir un coche sobre una pista de carreras, sin salirse de la carretera que está pintada en

negro.

```
10 MODE 0
20 GOSUB 1000
30 GOSUB 2000
40 END
1000 PAPER 12
1010 PEN 0
1020 CLS
1029 REM dibujo de la pista
1030 ladox=3:ladoy=7
1040 FOR y=ladoy TO ladoy+11
1050   LOCATE ladox,y
1060   PRINT CHR$(143)CHR$(143);
1070   LOCATE ladox+15,y
1080   PRINT CHR$(143)CHR$(143);
1090 NEXT
1100 xcom=7:ycom=5
1109 FOR con=-1 TO 1
1110   LOCATE xcom,ycom+con
1120   PRINT STRING$(8,CHR$(143));
1121   LOCATE xcom,ycom-1
1130   LOCATE xcom,ycom+15+con
1140   PRINT STRING$(8,CHR$(143));
1141 NEXT
1150 TAG
1160 color=0
1170 xizq=32:xder=576
1180 yabajo=150:yarriba=330
1190 ycambio=6:xcambio=32
1200 PLOT xizq+xcambio,yarriba+ycambio,color
1210 FOR con=1 TO 4
1220   yycambio=ycambio*con
1230   xxcambio=xcambio*con
1240   MOVE xizq+xxcambio,yarriba+yycambio
1250   PRINT CHR$(143);
1260   GOSUB 1600
1280   MOVE xder-xxcambio,yarriba+yycambio
1290   PRINT CHR$(143);
1300   GOSUB 1600
1320   MOVE xizq+xxcambio,yabajo-yycambio+8
1330   PRINT CHR$(143);
1340   GOSUB 1600
1360   MOVE xder-xxcambio,yabajo-yycambio+8
1370   PRINT CHR$(143);
1380   GOSUB 1600
```

```
1400 NEXT
1408 REM dos caracteres representando un coche
1409 REM uno para movimiento arriba/abajo; otro para movimiento
izquierda/derecha
1410 SYMBOL 240,0,102,36,126,126,36,102,0
1420 SYMBOL 241,0,90,126,24,24,126,90,0
1430 lado$=CHR$(240)
1440 arriba$=CHR$(241)
1450 coche$=lado$
1460 xcoche=400:ycoche=338
1470 PLOT xcoche+16,ycoche-4,3
1480 MOVE xcoche,ycoche
1490 PRINT coche$;
1500 RETURN
1600 FOR con1=1 TO 4
1610   MOVER -32,-16
1620   PRINT CHR$(143);
1630 NEXT
1640 RETURN
1999 REM examen del teclado
2000 golpe=0
2010 xcambio=0:ycambio=0
2020 WHILE golpe=0
2030   respuesta$=LOWER$(INKEY$)
2040   IF respuesta$="a" THEN xcambio=0:ycambio=2:coche$=arriba$:testx=-
16:testy=2
2050   IF respuesta$="z" THEN xcambio=0:ycambio=-2:coche$=arriba$:testx=-
16:testy=-16
2060   IF respuesta$="." THEN xcambio=4:ycambio=0:coche$=lado$:testx=0:testy=-8
2070   IF respuesta$="," THEN xcambio=-4:ycambio=0:coche$=lado$:testx=-
36:testy=-8
2079   REM la barra de espacio para parar el coche
2080   IF respuesta$=" " THEN xcambio=0:ycambio=0
2089   REM solo se dibuja el coche si se ha movido
2090   IF xcambio<>0 OR ycambio<>0 THEN GOSUB 3000
2100 WEND
2110 RETURN
2999 REM se comprueba el color del punto siguiente a la posicion actual
3000 color=TESTR(testx,testy)
3009 REM si no es el de ink 0 el coche esta fuera de la pista
3010 IF color<>0 THEN golpe=1:SOUND 7,500
3020 xcoche=xcoche+xcambio:ycoche=ycoche+ycambio
3030 MOVE xcoche,ycoche
3040 PRINT coche;
3050 RETURN
```



Ejercicios

1.Añada nuevos obstáculos a la pista de carreras, como balas de paja amarillas o los restos calcinados de un coche.

1.Cree una oruga verde, compuesta por varios caracteres, que se arrastre lentamente por la pantalla.

2.Añada frutas rojas situadas aleatoriamente en la pantalla; cuando la oruga coma alguna de ellas, debe volverse azul y morirse.

2.Programe el control de la oruga mediante el teclado, y ayúdela a salvarse de los peligros moviéndola verticalmente para evitar las frutas. El animal quedará a salvo cuando consiga alcanzar un paquete de hierba verde situado a la derecha de la pantalla.

3. Gráficas y diagramas

En el capítulo anterior nos hemos interesado por la manera en que se pueden emplear las técnicas gráficas en los juegos; pero, incluso en un microordenador, podemos encontrar muchas aplicaciones serias para estas técnicas. En los últimos años han proliferado los programas para pequeñas empresas, programas que llegan a tener un alto nivel de sofisticación. Muchos de ellos tienen como objetivo primordial la presentación más adecuada de series de datos para facilitar su rápida comprensión. En lugar de proporcionar largas listas llenas de cifras, lo que hacen es estructurar todos esos datos en forma de gráficas, diagramas de barras, diagramas de sectores, etc., y de combinaciones de estos tipos. El color y las gráficas de alta resolución hacen que sea más sencillo reconocer las características esenciales o las tendencias en una serie de datos. El empleo del ordenador proporciona además la posibilidad de recalcular y dibujar con rapidez una nueva gráfica, para ilustrar, por ejemplo, las consecuencias de una disminución de los precios. Vamos a concentrarnos en este capítulo en las formas de programar la realización de las tres formas más usuales de representación de datos: gráficas, diagramas (o histogramas) de barras y diagramas de sectores. A estas alturas del libro es importante conocer ya algunas de las reglas básicas que deben respetarse cuando se desean desarrollar programas de aplicación.

En primer lugar, no hay que intentar escribir un programa como un todo. Es más fácil desarrollar, corregir y mejorar un programa que ha sido escrito en módulos, o sea, pequeñas secciones de programa cada una de las cuales tiene un propósito determinado, como, por ejemplo, dibujar los ejes en una gráfica o colorear una barra en un diagrama de barras.

En segundo lugar, un programa no es nada flexible si está ligado a valores específicos. Es posible que un programa para dibujar los ejes de una gráfica funcione perfectamente, pero, si contiene líneas como

```
100 MOVE 308,390
110 DRAW 308,120
120 DRAW 630,120
```

será difícil utilizarlo en otras circunstancias y, cuando haya que hacer otra gráfica diferente, habrá que reescribirlo completamente. Es mejor usar variables siempre que sea posible. Esto presenta varias ventajas: los nombres de variables pueden ser más significativos que las tiras de números y el programa será más fácil de adaptar o modificar al cabo de varios meses.

Además, el empleo de variables permite escribir un programa de uso general que proporcione la gráfica correspondiente a una serie de datos; a la hora de utilizarlo, los únicos cambios a hacer serán los propios datos, sin necesidad de reeditar el programa línea por línea para ponerlo de acuerdo con las nuevas circunstancias.

Escribir un programa de esta manera requiere inicialmente mayor reflexión y

más trabajo. Pero merece la pena el esfuerzo suplementario, ya que con ello se evita el escribir varios programas diferentes que realicen más o menos la misma tarea.

GRÁFICAS DE PUNTOS Y LÍNEAS

Cuando se va a realizar una gráfica, la primera cuestión a resolver es la de la resolución necesaria. En el Amstrad podemos elegir entre diferentes modos, con diferentes resoluciones horizontales y la misma vertical. En general, por su alta resolución, es mejor el modo 2 para dibujar una gráfica con muchos puntos. Pero, lamentablemente, en este modo sólo se pueden utilizar dos colores. Cuando el número de puntos a representar es inferior a 300, el modo 1 da un buen equilibrio entre resolución y color: 320 puntos en horizontal y cuatro colores.

El Amstrad está diseñado de forma que el cambio de modo de pantalla no afecte al intervalo de variación de las coordenadas. Por ello, los programas que siguen funcionarán generalmente bien en cualquiera de los modos, a pesar de variar la resolución.

Vamos a comenzar por desarrollar un programa (muy corto). Por el momento dedicaremos a la gráfica toda la pantalla, dejando el problema de la rotulación para más tarde. Escribiremos el programa como una serie de subrutinas. Con esto tendremos mayor flexibilidad para poder dibujar varias series de datos en la misma gráfica, o dibujar varias gráficas, sin necesidad de grandes cambios en el programa:

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ypunto=399
510 xpunto=639
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
550 minx=200
560 maxx=4000
570 difx=maxx-minx
580 miny=100
590 maxy=1000
600 dify=maxy-miny
610 puntox=difx/xpunto
620 puntoy=dify/ypunto
690 RETURN
799 REM leer puntos del data y dibujarlos
800 READ numdepuntos
805 DIM x(numdepuntos),y(numdepuntos)
```

```
810 FOR con=1 TO numdepuntos
815   READ x(con):xdib=(x(con)-minx)/puntox
820   READ y(con):ydib=(y(con)-miny)/puntoy
825   PLOT xdib,ydib
930 NEXT
990 RETURN
1000 DATA 5,200,100,1000,200,1500,300,2500,600,4000,1000
```



En las líneas 500 a 540 se dibujan los ejes. El origen se sitúa en la esquina inferior izquierda de la pantalla. Es importante conocer los valores máximo y mínimo de los datos que hay que representar, de manera que se pueda realizar la gráfica con una escala adecuada para que quepa en la pantalla. Estos valores se dan de manera explícita entre las líneas 550 a 600, aunque ya veremos más tarde cómo el ordenador puede obtener por sí mismo estos valores a partir de los datos a representar. Cuando el ordenador conoce la diferencia entre el máximo y el mínimo, puede calcular la longitud unitaria en el eje X y el eje Y para que quepan todos los datos; esto es lo que hace en las líneas 610 y 620. El origen representará entonces el punto (*min x*, *min y*), y la esquina superior derecha de la pantalla el punto (*max x*, *max y*). Finalmente, el programa lee los valores de las coordenadas x e y de los distintos datos, los reduce a escala y los dibuja en las líneas 800 a 830. Por el momento los datos están ordenados siguiendo el orden creciente para la coordenada x. Ya veremos más adelante qué hay que hacer cuando los datos están ordenados aleatoriamente.

Puede ejecutar varias veces el programa, cambiando en cada caso los valores de *max x* y de *max y* para ver el efecto que esto produce. Doblando el valor de *max x* el programa aplasta la gráfica hacia la izquierda, al mismo tiempo que permite utilizar mayores valores de x que los que figuran en los datos. Doblando el valor de *max y* se produce un aplastamiento de la gráfica hacia abajo. Cosas similares ocurren al cambiar *min x* y *min-y*. Si le disgusta el hecho de que el origen represente (100, 200) en lugar de (0, 0), cambie los valores de *min x* y *min y* a 0. (Observe que esto hace que en una gran zona de la pantalla no se dibujen

puntos.) Recuerde que los datos de la línea 1000 están en determinado intervalo, y que, cambiando los valores de *min x*, etc., puede perder puntos de la gráfica.

La ejecución del programa revela algunos problemas, como que los puntos y la gráfica son algo difíciles de apreciar. Algunos cambios permiten mejorar la situación:

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ypunto=399
510 xpunto=639
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
550 mina= 00
560 maxx=4000
570 difx=maxx-minx
580 miny=100
590 maxy=1000
600 dify=maxy-miny
610 puntox=difx/xpunto
620 puntoy=dify/ypunto
690 RETURN
799 REM leer puntos del data y dibujarlos
800 READ numdepuntos
805 DIM x(numdepuntos),y(numdepuntos)
806 READ tinta0,tinta1
807 INK 0,tinta0
808 INK 1,tinta1
809 PAPER 0:PEN 1
810 FOR con=1 TO numdepuntos
815   READ x(con):xdib=(x(con)-minx)/puntox
820   READ y(con):ydib=(y(con)-miny)/puntoy
825   PLOT xdib,ydib
830 NEXT
990 RETURN
1000 DATA 5,0,24,200,100,1000,200,1500,300,2500,600,4000,1000
```



Ahora, los colores que se utilizan en la gráfica se especifican en la sentencia DATA. De la misma forma podríamos especificar MODE, pero vamos a mantenernos en el modo 1. Podemos hacer más visibles los puntos dibujando en su lugar una cruz o un cuadrado. Esto es muy sencillo de hacer utilizando movimientos relativos:

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ypunto=399
510 xpunto=639
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
550 minx=200
560 maxx=4000
570 difx=maxx-minx
580 miny=100
590 maxy=1000
600 dify=maxy-miny
610 puntox=difx/xpunto
620 puntoy=dify/ypunto
690 RETURN
799 REM leer puntos del data y dibujarlos
800 READ numdepuntos
805 DIM x(numdepuntos),y(numdepuntos)
806 READ tinta0,tinta1
807 INK 0,tinta0
808 INK 1,tinta1
```

```
809 PAPER 0:PEN 1
810 cruzx=10
811 cruzy=10
814 FOR con=1 TO numdepuntos
815   READ x(con):xdib=(x(con)-minx)/puntox
820   READ y(con):ydib=(y(con)-miny)/puntoy
825   PLOT xdib,ydib
830   MOVER -cruzx,cruzy
840   DRAWR 2*cruzx,-2*cruzy
850   MOVER -2*cruzx,0
860   DRAWR 2*cruzx,2*cruzy
870   MOVER -cruzx,-cruzy
880 NEXT
990 RETURN
1000 DATA 5,0,24,200,100,1000,200,1500,300,2500,600,4000,1000
```

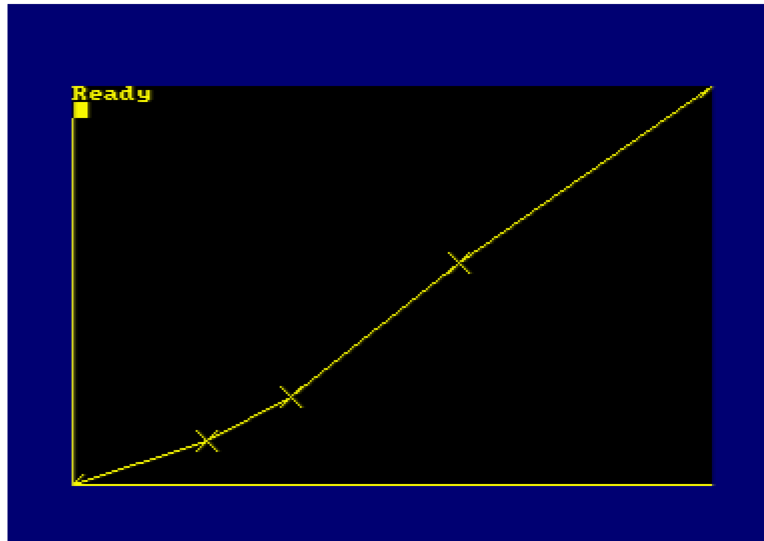
Los datos para los brazos de la cruz se fijan mediante dos variables, en lugar de tomarlos de la sentencia DATA. Es mejor así, ya que serán datos que no cambiarán de una a otra ejecución del programa. Puede modificar las longitudes para que la cruz sea mayor o menor que la del ejemplo; bastará con cambiar dos líneas.



El programa mejorará si damos la opción de unir los sucesivos puntos de la gráfica. Para ello se utiliza la variable unir que, según el valor que tome, hace que los puntos se dibujen unidos o sueltos:

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ypunto=399
```

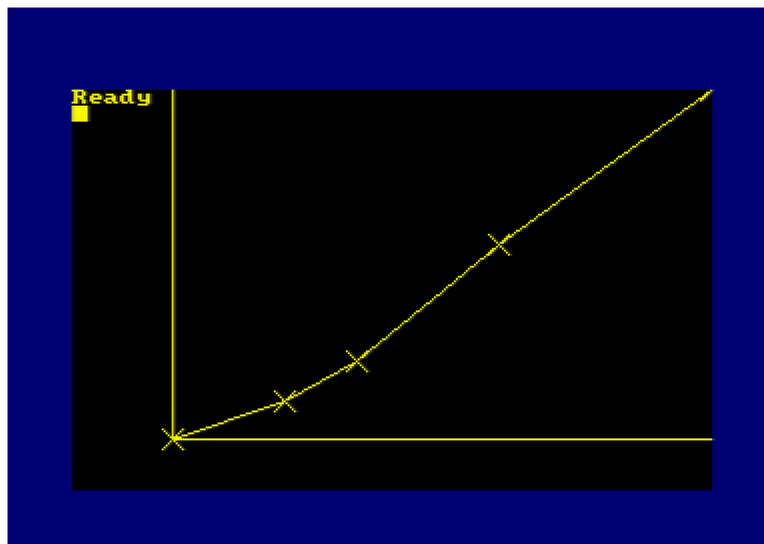
```
510 xpunto=639
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
550 minx=200
560 maxx=4000
570 difx=maxx-minx
580 miny=100
590 maxy=1000
600 dify=maxy-miny
610 puntox=difx/xpunto
620 puntoy=dify/ypunto
690 RETURN
799 REM leer puntos del data y dibujarlos
800 READ numdepuntos
805 DIM x(numdepuntos),y(numdepuntos)
806 READ tinta0,tinta1
807 INK 0,tinta0
808 INK 1,tinta1
809 PAPER 0:PEN 1
810 cruzx=10
811 cruzy=10
812 unir=1
814 FOR con=1 TO numdepuntos
815 READ x(con):xdib=(x(con)-minx)/puntox
816 x(con)=xdib
820 READ y(con):ydib=(y(con)-miny)/puntoy
821 y(con)=ydib
825 PLOT xdib,ydib
830 MOVER -cruzx,cruzy
840 DRAWR 2*cruzx,-2*cruzy
850 MOVER -2*cruzx,0
860 DRAWR 2*cruzx,2*cruzy
870 MOVER -cruzx,-cruzy
874 REM si unir esta a 1 los puntos se van uniendo
875 IF unir=1 AND con>1 THEN DRAW x(con-1),y(con-1)
880 NEXT
990 RETURN
1000 DATA 5,0,24,200,100,1000,200,1500,300,2500,600,4000,1000
```



El mayor defecto de este programa es la falta de rotulación. Como el gráfico está pegado a la esquina inferior izquierda de la pantalla, no hay sitio para ella. Puede parecer que esto obliga a modificaciones importantes del programa. De hecho, el uso de variables y la capacidad del Amstrad para cambiar el origen hacen que sea muy sencillo mover los ejes:

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ypunto=399
510 xpunto=639
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
550 minx=200
560 maxx=4000
570 difx=maxx-minx
580 miny=100
590 maxy=1000
600 dify=maxy-miny
610 puntox=difx/xpunto
620 puntoy=dify/ypunto
690 RETURN
799 REM leer puntos del data y dibujarlos
800 READ numdepuntos
805 DIM x(numdepuntos),y(numdepuntos)
806 READ tinta0,tinta1
807 INK 0,tinta0
808 INK 1,tinta1
809 PAPER 0:PEN 1
```

```
810 cruzx=10
811 cruzy=10
812 unir=1
814 FOR con=1 TO numdepuntos
815   READ x(con):xdib=(x(con)-minx)/puntox
816   x(con)=xdib
820   READ y(con):ydib=(y(con)-miny)/puntoy
821   y(con)=ydib
825   PLOT xdib,ydib
830   MOVER -cruzx,cruzy
840   DRAWR 2*cruzx,-2*cruzy
850   MOVER -2*cruzx,0
860   DRAWR 2*cruzx,2*cruzy
870   MOVER -cruzx,-cruzy
874   REM si unir esta a 1 los puntos se van uniendo
875   IF unir=1 AND con>1 THEN DRAW x(con-1),y(con-1)
880 NEXT
990 RETURN
1000 DATA 5,0,24,200,100,1000,200,1500,300,2500,600,4000,1000
```



Hemos cambiado las antiguas líneas 500 y 510, puesto que ya no utilizamos toda la pantalla para la gráfica. El resto del programa puede continuar igual; las líneas y puntos se dibujarán ahora en relación al nuevo origen, como comprobará ejecutando el programa. Si cambia los valores de ox y oy, verá cómo no cambia el aspecto de la curva al trasladar el origen, salvo en lo que se refiere a su tamaño.

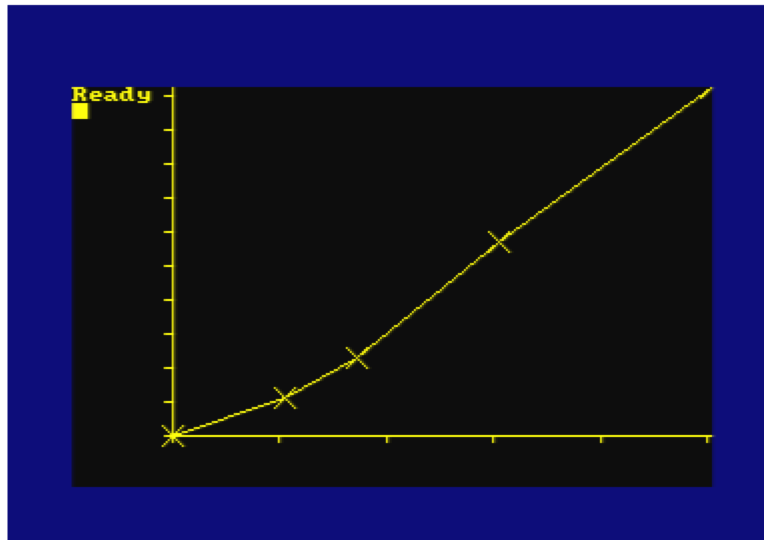
Ahora tenemos sitio para poner marcas en los ejes y escribir los correspondientes rótulos; pero esto no se puede hacer de manera totalmente automática. Seguramente no tendremos problemas con el eje y, ya que parece que, al elegir el origen, hemos dejado sitio suficiente para los rótulos.

Por otro lado, aunque tengamos sitio en el eje x para 10 divisiones, nada impide que consideremos más conveniente hacer un número menor, como, por

ejemplo, 5 divisiones. El número máximo de divisiones está sin embargo limitado por dos factores: la resolución del modo que se emplee y la anchura de los caracteres que vayan a escribirse debajo de las marcas. Lo que el ordenador puede hacer es calcular en qué casos vamos a obtener rótulos que se solapen unos con otros; pero no puede elegir por sí mismo un intervalo menor que sea conveniente. Los números que representan distancias adecuadas, como 0.5, 10, 20, 25, 100, etc., dependen más bien de nuestros hábitos y de las circunstancias. Vamos a hacer que seamos nosotros quienes elijamos los intervalos y que el ordenador rechace los valores que no sean razonables. El etiquetado se divide en dos partes: poner las marcas y escribir los rótulos. La primera acción puede no ser correcta si conduce a que los rótulos se solapen. Hay que informar al ordenador de la máxima longitud de los rótulos que deben escribirse, para que pueda calcular si podrá o no colocarlos uno tras otro debajo del eje X:

```
503 REM se oculta la grafica hasta que este completa
504 INK 0,24:INK 1,24
628 REM numero de marcas en los ejes x e y
629 REM sin incluir la del origen
630 xcantidad=5
640 ycantidad=10
649 REM distancia grafica entre las marcas
650 xancho=INT(xpunto/xcantidad)
660 yalto=INT(ypunto/ycantidad)
668 REM cantidad maxima de caracteres de los numeros del eje x
669 REM esto tendra que cambiarlo para adaptarlo a sus propios datos
670 maxxcadena=4
673 REM tamano de los caracteres en el modo 1 (en numero de puntos)
674 REM el ancho debe Cambiar a 32 para el modo 0 y a 8 para el modo 2
675 charancho=16
676 charalto=16
679 REM se calcula la maxima longitud de la cadena en puntos graficos
680 grafxcadena=charancho*maxxcadena
688 REM no se rotulan los ejes cuando habria que poner numeros demasiado grandes
689 REM o cuando la distancia entre marcas es menor que el temario de un
caracter
690 IF xancho<grafxcadena OR xancho<charancho THEN RETURN
691 REM lo mismo para el eje y
692 maxycadena=4
694 grafycadena=charancho*maxycadena
696 IF ox<grafycadena OR yalto<charalto THEN RETURN
699 REM longitud de las marcas de los ejes
700 xmarca=6
702 ymarca=8
703 REM el rotulo de cada marca sera de un valor xvalor mas alto que el anterior
704 xvalor=difx/xcantidad
706 TAG
707 REM se avanza x cantidad veces para obtener el numero de marcas que
```

```
corresponde
708 FOR con=0 TO xcantidad
710   MOVE 0,0
711   REM avance en el eje hasta en comienzo de la marca
712   MOVE xancho*con,0
713   REM dibujo de la marca
714   DRAWR 0,-xmarca
728 NEXT
729 REM lo mismo para el eje y
730 yvalor=dify/ycantidad
732 FOR con=0 TO ycantidad
734   MOVE 0,0
736   MOVER 0,yalto*con
738   DRAWR -ymarca,0
754   NEXT
790 RETURN
```

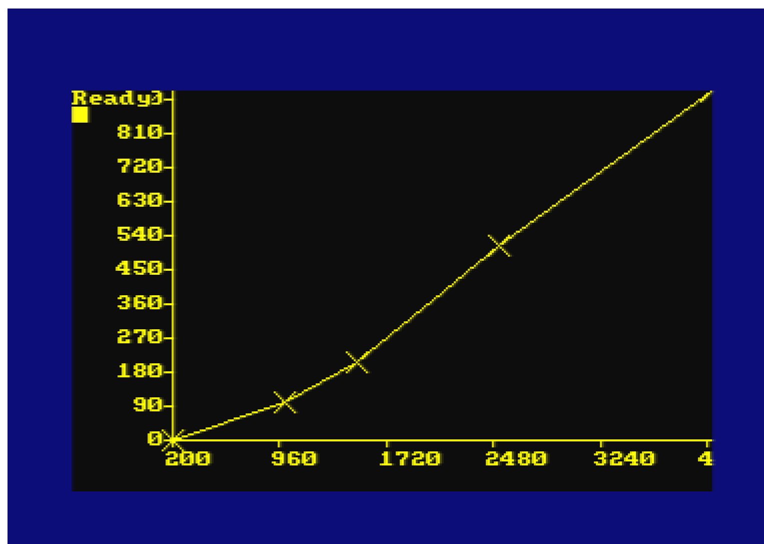


Si no le gusta el tipo de marcas que hemos elegido, puede usted programar el que desee. Para ello basta con que modifique las líneas correspondientes.

En lugar de volver a calcular las posiciones de las marcas en el momento de poner los rótulos, lo mejor es incluir una rutina que ponga el texto al mismo tiempo que las marcas:

```
708 FOR con=0 TO xcantidad
710   MOVE 0,0
711   REM avance en el eje hasta en comienzo de la marca
712   MOVE xancho*con,0
713   REM dibujo de la marca
714   DRAWR 0,-xmarca
715   REM posicion para escribir el rotulo de la marca
716   MOVER -charancho/2,-xmarca
718   numero$=STR$(minx+con*xvalor)
```

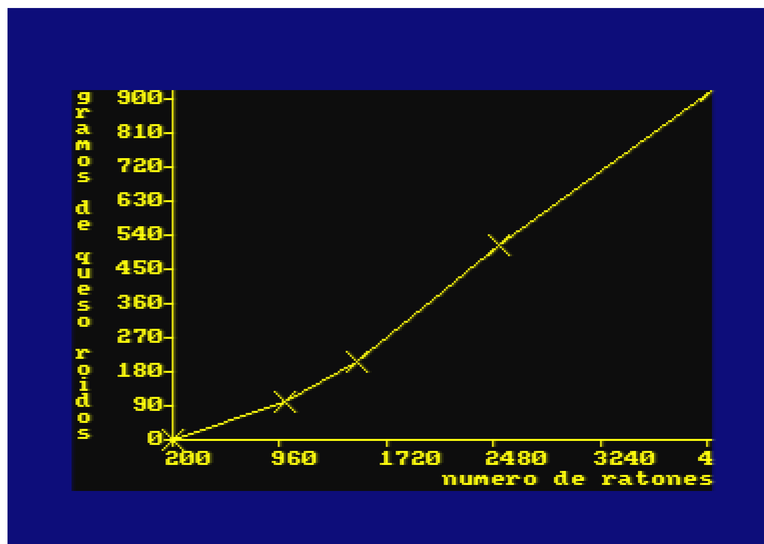
```
719   REM se trunca el numero si es demasiado largo
720   numero$=MID$(numero$,2,maxxcadena)
722   longitud=LEN(numero$)
723   REM se suprime el punto decimal si es el ultimo caracter
724   IF MID$(numero$,longitud)="." THEN numero$=MID$(numero$,1,longitud-1)
726   PRINT numero$;
728 NEXT
729 REM lo mismo para el eje y
730 yvalor=dify/ycantidad
732 FOR con=0 TO ycantidad
734 MOVE 0,0
736 MOVER 0,yalto*con
738 DRAWR -ymarca,0
740 MOVER -grafycadena,charalto/2
742   numero$=STR$(minyl-con*yvalor)
744   numero$=MID$(numero$,2,maxycadena)
746   longitud=LEN(numero$)
748   IF MID$(numero$,langitud)="." THEN numero$=MID$(numero$,1,longitud-1)
749   REM esto es para alinear los numeros junto a las marcas
750   IF LEN(numero$)<maxycadena THEN numero$=STRING$(maxycadena-LEN(numero$),"
")+numero$
752   PRINT numero$;
754 NEXT
790 RETURN
```



También podemos poner un rótulo en cada uno de los ejes y escribir un título general de la gráfica. Para ello el origen debe estar elegido de manera que quede sitio para todo esto.

```
755 REM rotulos de los ejes
```

```
756 xrotulo$="numero de ratones"
758 yrotulo$="gramos de queso roidos"
759 REM longitud del rotulo en puntos graficos
760 xrotulolon=LEN(rotulo$)*charancho
761 REM espacio antes del rotulo, para que quede centrado
762 xrotulocom=(xpunto-xrotulolon)/2
763 REM colocarse a la altura conveniente debajo del eje x para escribir el rotulo
764 MOVE xrotulocom,-2*charalto
766 PRINT xrotulo$;
767 REM lo mismo para el eje y
768 yrotulolon=LEN(yrotulo$)*charalto
770 yrotulocom=(ypunto+yrotulolon)/2
771 REM hay que escribir separadamente cada letr a del rotulo, ya que va en vertical
772 FOR con=1 TO LEN(yrotulo$)
773   REM se extrae un caracter del rotulo
774   char$=MID$(yrotulo$,con,1)
775   REM ponerse convenientemente a la izquierda para imprimir el caracter
776   MOVE -charancho*(maxycadena+2),yrotulocom-charalto*(con-1)
778   PRINT char$;
780 NEXT
790 RETURN
```

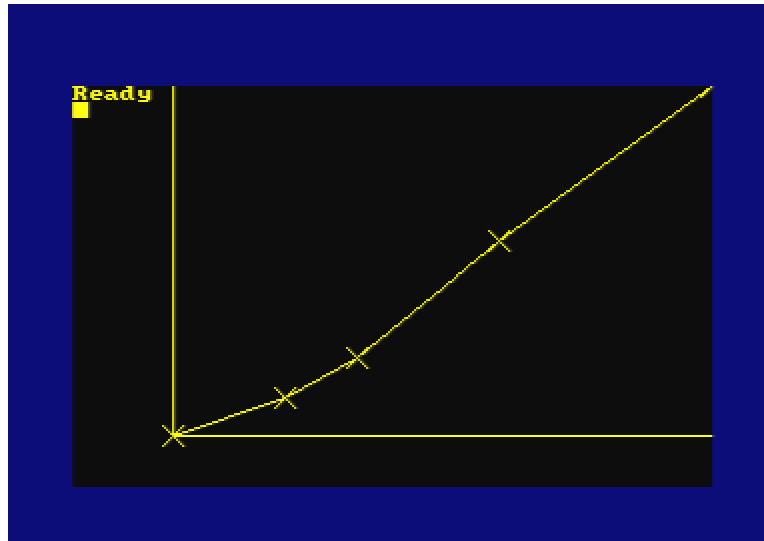


Todavía no está completo el programa, ya que sólo puede trabajar con datos previamente ordenados. En su forma actual puede ser útil para muchas cosas, como por ejemplo para mediciones de la precipitación de lluvia en un período de tiempo, o el estado de su cuenta bancaria mes a mes (aunque esto puede obligarle a poner un eje negativo). Cuando los datos no están ordenados de esta manera, hay que hacer que el ordenador los vaya leyendo y reordenando en caso necesario. Para ello se almacenan los datos en un vector y se procede a su

clasificación:

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ox=100:oy=50
505 ORIGIN ox,oy
510 ypunto=399-oy
515 xpunto=639-ox
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
550 minx=200
560 maxx=4000
570 difx=maxx-minx
580 miny=100
590 maxy=1000
600 dify=maxy-miny
610 puntox=difx/xpunto
620 puntoy=dify/ypunto
690 RETURN
799 REM leer puntos del data y dibujarlos
800 READ numdepuntos
805 DIM x(numdepuntos),y(numdepuntos)
806 READ tinta0,tinta1
807 INK 0,tinta0
808 INK 1,tinta1
809 PAPER 0:PEN 1
810 cruzx=10
811 cruzy=10
812 unir=1
814 FOR con=1 TO numdepuntos
815   READ x(con):xdib=(x(con)-minx)/puntox
816   x(con)=xdib
820   READ y(con):ydib=(y(con)-miny)/puntoy
821   y(con)=ydib
825 NEXT
829 REM ordenar la matriz x
830 FOR con= 2 TO numdepuntos
840   FOR valor=con TO 2 STEP -1
850     IF x(valor)<x(valor-1) THEN GOSUB 1500 ELSE valor=2
860   NEXT
870 NEXT
880 FOR con=1 TO numdepuntos
890   PLOT x(con),y(con)
```

```
900    MOVER -cruzx,cruzy
910    DRAWR 2*cruzx,-2*cruzy
920    MOVER -2*cruzx,0
930    DRAWR 2*cruzx,2*cruzy
940    MOVER -cruzx,-cruzy
950    REM si unir esta a 1 los puntos se van uniendo
960    IF unir=1 AND con>1 THEN DRAW x(con-1),y(con-1)
970    NEXT
990 RETURN
999 REM los datos estan ahora desordenados
1000 DATA 5,0,24,1000,200,4000,1000,200,100,2500,600,1500,300
1499 REM cambio en las matrices para que la coordenada x mas baja vaya primero
1500 tempx=x(valor):tempy=y(valor)
1510 x(valor)=x(valor-1):y(valor)=y(valor-1)
1520 x(valor-1)=tempx:y(valor-1)=tempy
1530 RETURN
```



El método de clasificación que se ha utilizado es el de inserción. Se ordenan las dos primeras coordenadas x; luego se coloca la tercera en su posición correcta respecto de las otras dos. El proceso se repite con la cuarta y así sucesivamente, hasta que todas quedan colocadas de menor a mayor

La clasificación de datos numéricos o literales es uno de los temas más importantes en informática. Es posible que usted encuentre muy lento e procedimiento anterior si debe ordenar varios centenares de datos, y que prefiera otro método. El método de ordenación de burbuja es muy utilizado cuando se tienen datos que ya están algo ordenados. Sin embargo, para clasificación de grandes cantidades de datos, la única forma de conseguir métodos rápidos es programarlos en código de máquina.

La ventaja de clasificar los datos es que al mismo tiempo se pueden calcular los valores de min x, min y, max x y max y.

El programa ha quedado con ciertos puntos débiles que usted podrá rectificar en los ejercicios que siguen. Sin embargo, es adecuado para bastantes situaciones.

Ejercicios

1. ¿Qué cambios hay que hacer para que el programa se pueda ejecutar en los modos 0 o 2? ¿Cómo reemplazar por variables los valores que dependen específicamente del modo de pantalla?
2. En el programa, el eje Y se dibuja siempre a la izquierda y el eje X abajo. Modifique el programa para que admita valores positivos y negativos de x y de y y para que trace los ejes por el punto (0, 0).
3. Modifique el programa para que admita simultáneamente varios conjuntos de datos y los represente con colores diferentes. Hágalo sin repetir secciones del programa; llame como subrutinas a las partes del actual programa que necesite.
4. Modifique el programa para que dibuje dos gráficas con escalas diferentes en las dos mitades superior e inferior de la pantalla. (Necesitará cambiar el valor de los límites de la variable y y cambiar dos veces el origen.)
5. Mejore el programa para que los datos puedan ser almacenados y leídos de un fichero.

DIAGRAMAS DE BARRAS

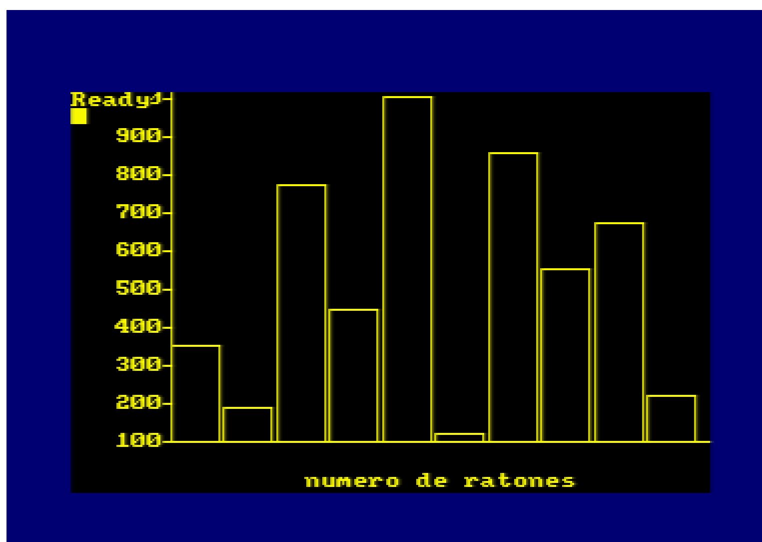
Ahora que disponemos de un programa para realizar gráficas, nos resultará bastante sencillo dibujar diagramas de barras, ya que hemos resuelto en la sección anterior muchos de los problemas que se presentan. El proceso se simplifica bastante al tener que ocuparse sólo de la coordenada Y; en el otro eje es muy fácil calcular la anchura de las barras. Además, los datos ya están ordenados, con lo que no hay que atender a su clasificación. Las semejanzas con el programa precedente ponen de relieve las ventajas del uso sistemático de variables y subrutinas.

```
10 MODE 1
20 GOSUB 500
30 GOSUB 800
40 END
499 REM dibujar los ejes
500 ox=100:oy=50
503 REM se oculta la grafica hasta que este completa
504 INK 0,24:INK 1,24
505 ORIGIN ox,oy
```

```
510 ypunto=399-oy
515 xpunto=639-ox
520 MOVE 0,ypunto
530 DRAW 0,0,1
540 DRAW xpunto,0
580 miny=100
590 maxy=1000
600 dify=maxy-miny
620 puntoy=dify/ypunto
628 REM numero de marcas; esta vez solo en el eje y
629 REM xcantidad es el numero de barras
630 READ numerodebarras:xcantidad=numerodebarras
640 ycantidad=9
649 REM distancia grafica entre las marcas
650 xancho=INT(xpunto/xcantidad)
660 yalto=INT(ypunto/ycantidad)
673 REM tamano de los caracteres en el modo 1 (en numero de puntos)
674 REM el ancho debe cambiar a 32 para el modo 0 y a 8 para el modo 2.
675 charancho=16
676 charalto=16
692 maxycadena=4
694 grafycadena=charancho*maxycadena
696 IF ox<grafycadena OR yalto<charalto THEN RETURN
699 REM longitud de las marcas de los ejes
700 xmarca=6
702 ymarca=8
706 TAG
728 REM el rotulo de cada marca sera de un valor yvalor mas alto que el anterior
729 REM se avanza ycantidad veces para obtener el numero de marcas que
corresponde
730 yvalor=dify/ycantidad
732 FOR con=0 TO ycantidad
734   MOVE 0,0
736   MOVER 0,yalto*con
738   DRAWR -ymarca,0
740   MOVER -grafycadena,charalto/2
742   numero$=STR$(miny+con*yvalor)
744   numero$=MID$(numero$,2,maxycadena)
746   longitud=LEN(numero$)
748   IF MID$(numero$,longitud)="." THEN numero$=MID$(numero$,1,longitud-1)
749   REM esto es para alinear los numeras junto a las marcas
750   IF LEN(numero$)<maxycadena THEN numero$=STRING$(maxycadena-LEN(numero$),"
")+numero$
752   PRINT numero$;
754 NEXT
755 REM rotulos de los ajes
756 xrotulo$="numero de ratones"
```

```
758 yrotulo$="gramos de queso roidos"
759 REM longitud del rotulo en puntos graficos
760 xrotulolon=LEN(xrotulo$)*charancho
761 REM espacio antes del rotulo, para que quede centrado
762 xrotulocom=(xpunto-xrotulolon)/2
763 REM colocarse a la altura conveniente debajo del eje x para escribir el
rotulo
764 MOVE xrotulocom,-2*charalto
766 PRINT xrotulo$;
767 REM lo mismo para El eje y
768 yrotulolon=LEN(yrotulo$)*charalto
770 yrotulocom=(ypunto+yrotulolon)/2
771 REM hay que escribir separadamente cada letra del rotulo, ya que va en
vertical
772 FOR con=1 TO LEN(yrotulo$)
773   REM se extrae un caracter del rotulo
774   char$=MID$(yrotulo$,con,1)
775   REM ponerse convenientemente a la izquierda para imprimir el caracter
776   MOVE -charancho*(maxycadena+2),yrotulocom-charalto*(con-1)
770   PRINT char$;
780 NEXT
790 RETURN
799 REM leer puntos del data y dibujar las barras
800 DIM y(numerodebarras)
806 READ tinta0,tinta1
807 INK 0,tinta0
808 INK 1,tinta1
809 PAPER 0:PEN 1
814 FOR con=1 TO numerodebarras
815   READ y
816   x(con)=xdib
819   REM altura de la barra calculada a la escala conveniente
820   y(con)=(y-miny)/puntoy
825 NEXT
829 REM reducir la anchura de las barras para dejar espacio entre ellas
830 barraancho=xancho-4
840 FOR con=1 TO numerodebarras
850   MOVE 0,0
860   DRAWR 0,y(con),1
870   DRAWR barraancho,0
880   DRAWR 0,-y(con)
890   ox=ox+xancho
899   REM desplazamiento del origen para dibujar la barra siguiente
900   ORIGIN ox,oy
910 NEXT
990 RETURN
999 REM ahora solo se necesita la altura de las barras; la anchura es fija
```

```
1000 DATA 10,0,24,350,190,760,440,990,124,846,545,666,222
```



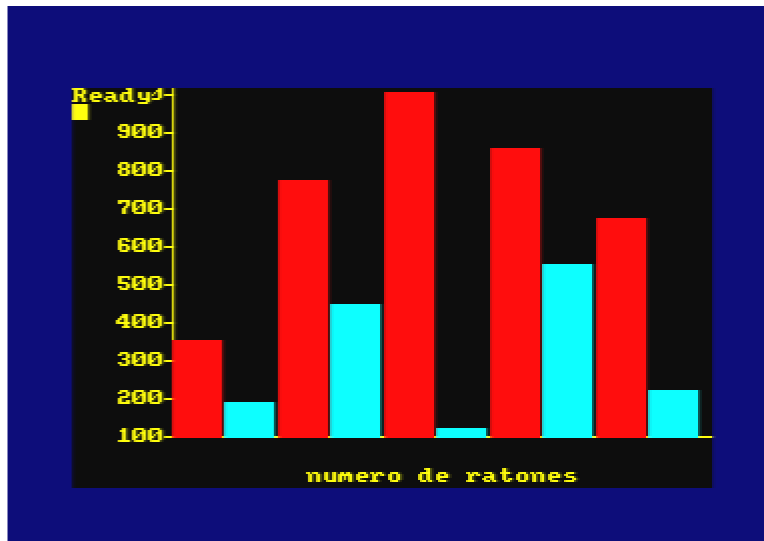
La subrutina 800 dibuja cada barra mediante una serie de movimiento: referidos al origen. Desplazando este origen una distancia fija cada vez que se dibuja una barra, se puede utilizar un bucle para dibujar todas las barras.

El dibujo queda mejor si se rellenan las barras con diferentes colores

Lamentablemente, el Amstrad CPC464[1] no tiene un comando que rellene con un color un área determinada. Hay que rellenar las barras dibujando líneas sencillas de la manera más rápida posible. En el último capítulo veremos algunos procedimientos para acelerar un programa, pero ya podemos emplear aquí algunos de esos conocimientos:

```
830 barraancho=xancho-4
835 color=2
840 FOR con=1 TO numerodebarras
844   REM se alterna el color de las barras
845   IF color=3 THEN color=2 ELSE color=3
848   REM las barras se rellenan mas rapidamente t omando un paso
849   REM adecuado a la resolucion del modo de pan talla, para no repetir lineas
850   FOR barra=0 TO barraancho STEP charancho/8
860     MOVE 0+barra,0
870     DRAWR 0,y(con),color
880   NEXT
890   ox=ox+xancho
899   REM desplazamiento del origen para dibujar la siguiente barra
900   ORIGIN ox,oy
910 NEXT
990 RETURN
```

1 Los modelos posteriores incluyen la orden FILL (“rellenar”).

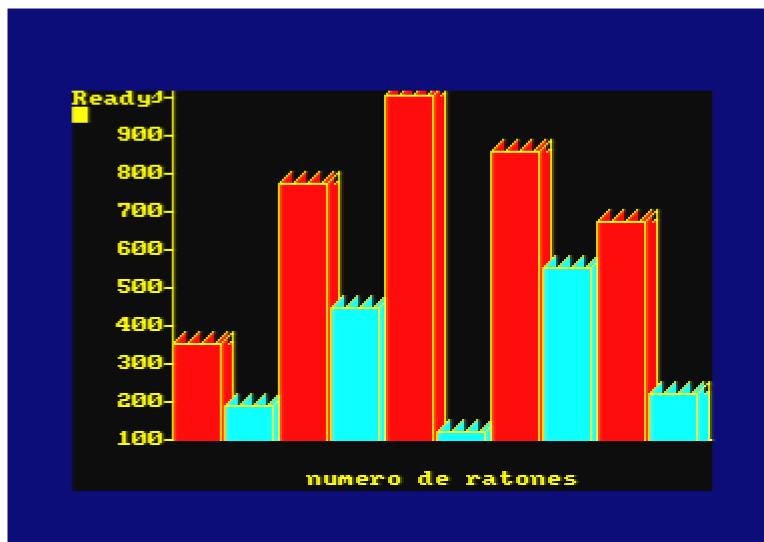


Las líneas se dibujan verticalmente en lugar de horizontalmente, ya que muchas barras serán más altas que anchas y se reduce así el número de líneas a dibujar. Pero rellenados de color realmente rápidos sólo pueden realizarse con rutinas en código de máquina.

Se obtiene una interesante variación sobre los diagramas de barras dibujando éstas con aspecto sólido, lo que da a la figura un aire tridimensional:

```
830 barraancho=xancho-4
831 REM barralado es la anchura del lateral de la barra
832 REM barrasuple es la altura suplementaria de la parte trasera de la barra
sobre la delantera
833 REM modifique los valores segun quiera que a parezcan mas o menos profundas
834 barralado=barraancho/4:barrasuple=barralado
835 color=2
840 FOR con=1 TO numerodebarras
844   REM se alterna el color de las barras
845   IF color=3 THEN color=2 ELSE color=3
846   barrasuplecon=0
848   REM tambien hay que rellenar el lateral de 1 a barra
849   REM luego el ancho de la barra es barraanchq 1-barralado
850   FOR barra=0 TO barraancho+barralado STEP charancho/8
859     REM el fin de la linea se dibuja en otro color como ocurrira con todas
las aristas de la bar ra
860     PLOT 0+barra,0,1
869     REM la linea hasta arriba de la barra
870     DRAWR 0,y(con)+barrasuplecon,color
871     REM el otro final de la linea tambien en color diferente
872     PLOTR 0,0,1
873     REM la linea siguiente debe ser un poco mas alta
874     REM y asi hasta alcanzar la parte de atras de la barra
875     barrasuplecon=barrasuplecon+charancho/8:IF barrasuplecon>barrasuple THEN
```

```
barrasuplecon=bartop
880    NEXT
881    REM las aristas de la barra van en otro color
882    y=y(con):MOVE 0,0
883    DRAWR 0,y,1
884    DRAWR barraancho,0
885    DRAWR 0,-y
886    MOVER 0,y
887    DRAWR barralado,barrasuple
888    DRAWR 0,-y-barrasuple
890    ox=ox+xancho
899    REM desplazamiento del origen para dibujar l a siguiente barra
900    ORIGIN ox,oy
910 NEXT
990 RETURN
```



Ejercicios

1. Modifique el programa para que funcione correctamente en modo 0 y en modo 2.
2. Modifique el programa para que las barras del diagrama sean horizontales en lugar de verticales.
3. Cambie el diagrama en 3D para que se pueda representar una sucesión de diagramas, cada uno al lado del anterior.

DIAGRAMAS DE SECTORES

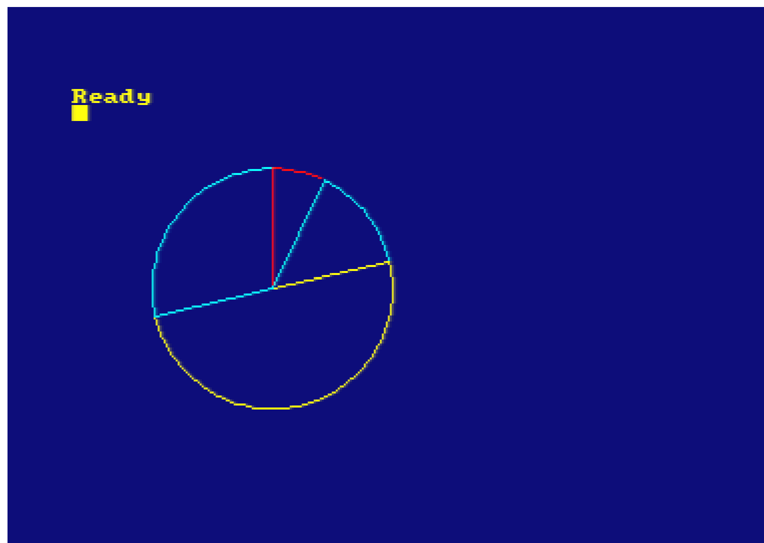
La realización de un diagrama de sectores (o sea, un histograma consistente en un círculo o un pequeño cilindro dividido en sectores de tamaño proporcional a la

magnitud de los datos) tiene poco en común con el dibujo de gráficas y de diagramas de barras. Por eso vamos a diseñar un programa completamente nuevo, aunque conservando algunas características generales de los anteriores, como el empleo sistemático de variables y subrutinas. Para este diseño es importante la resolución, puesto que necesitamos dibujar una circunferencia con bastante exactitud. Pero también lo es el color, así que vamos a tomar una decisión de compromiso y a elegir el modo 1.

El programa siguiente construye uno de estos diagramas, dividiendo un círculo en sectores de tamaño adecuado, cada uno con su contorno coloreado:

```
10 MODE 1
20 GOSUB 1000
30 GOSUB 2000
40 GOSUB 3000
50 END
997 REM leer del DATA las coordenadas del centro y el radio
1000 READ centrox,centroy
1010 READ radio
1020 READ numdeval
1070 DIM valor(numdeval),angulo(numdeval)
1039 REM sumar todos los valores para calcular 1 as divisiones del circulo
1040 totalvalor=0
1050 FOR con=1 TO numdeval
1060   READ valor(con)
1070   totalvalor=totalvalor+valor(con)
1080 NEXT
1090 RETURN
1100 DATA 200,200,120,4,1,2,7,4
1999 REM calculo del angulo de cada sector
2000 FOR con=1 TO numdeval
2010   angulo(con)=2*PI*valor(con)/totalvalor
2020 NEXT
2030 RETURN
3000 REM cambiaremos esto enseguida
3010 comangulo=0
3020 incremento=PI/60
3030 color=1
3039 REM para modo 0 hacer numdecol=15 (sin contar el color del fondo)
3040 numdecol=3
3050 FOR con=1 TO numdeval
3057   REM angulo en que termina el sector
3060   finangulo=comangulo+angulo(con)
3069   REM color del sector
3070   color=1+(color+1) MOD numdecol
3079   REM asegurarse de que el primero y el ultimo sector son de colores
diferentes
```

```
3080   IF con=numdeval AND numdeval MOD numdecol=1 THEN color=1+(color+1) MOD
numdecol
3090   MOVE centrox,centroy:DRAW
centrox+radio*SIN(comangulo),centroy+radio*COS(comangulo),color
3099   REM dibujo del sector
3100   FOR angulo=comangulo TO finangulo STEP incremento
3110     DRAW centrox+radio*SIN(angulo),centroy+radio*COS(angulo)
3120   NEXT
3129   REM angulo de comienzo del nuevo sector
3130   comangulo=finangulo
3140 NEXT
3150 RETURN
```



La subrutina 1000 lee de líneas DATA los valores que hay que representar y calcula su suma. La proporción entre un valor y la suma total es lo que da el ángulo que ocupa el correspondiente sector; este ángulo se calcula con la rutina 2000. La rutina 3000 dibuja cada sector con su color.

Al ejecutar el programa verá que, si el radio del diagrama es grande, el dibujo es lento. Algo se puede ganar en velocidad si se toma como nuevo origen el centro del círculo:

```
3000 ORIGIN centrox,centroy
3090 MOVE 0,0:DRAW radio*SIN(comangulo),radio*COS(comangulo),color
3099 REM dibujo del sector
3100 FOR angulo=comangulo TO finangulo STEP incremento
3110   DRAW radio*SIN(angulo),radio*COS(angulo)
3120 NEXT
```

Los cálculos más largos son, no obstante, los de los senos y cosenos, y esto, a diferencia de lo que ocurre con otros cálculos, no se puede mejorar con el empleo de números enteros. Vamos a modificar algunas cosas para minimizar estos

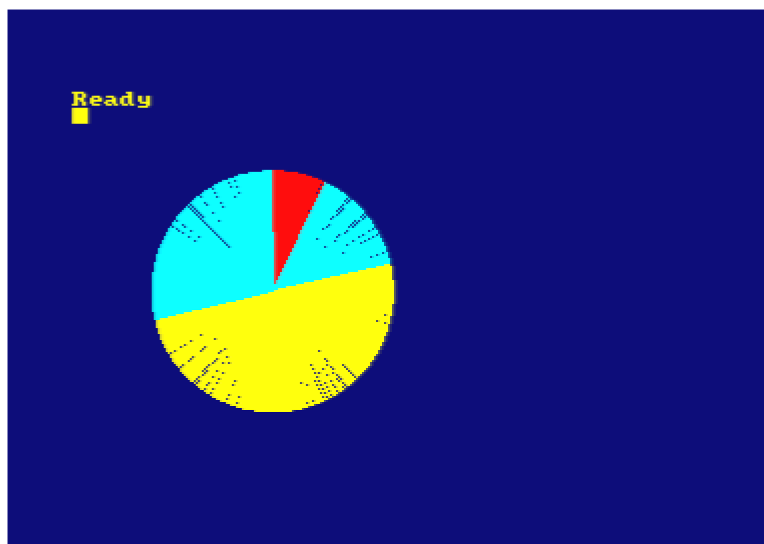
cálculos; la idea puede servir para objetivos distintos del que nos ocupa. Ahora sólo se calcula un seno y un coseno:

```
2000 radval=radio*4
2005 FOR con=1 TO numdeval
2010   angulo(con)=radval*valor(con)/totalvalor
2020 NEXT
2030 RETURN
3000 ORIGIN centrox,centroy
3009 REM al emplear enteros en el bucle aumenta la velocidad
3010 DEFINT c
3030 color=1
3035 elseno=SIN(2*PI/radval):elcoseno=COS(2*PI/radval)
3037 x1=radio:y1=0
3039 REM para modo 0 hacer numdecol=15 (sin cont ar el color del fondo)
3040 numdecol=3
3050 FOR con=1 TO numdeval
3069   REM color del sector
3070   color=1+(color+1) MOD numdecol
3079   REM asegurarse de que el primero y el ultim o sector son de colores
diferentes
3080   IF con=numdeval AND numdeval MOD numdecol=1 THEN color=1+(color+1) MOD
numdecol
3090   REM
3099   REM dibujo del sector
3100   FOR con1=1 TO angulo(con)
3110     x=x1*elcoseno-y1*elseno
3112     y=x1*elseno+y1*elcoseno
3114     PLOT x,y,color
3116     x1=x:y1=y
3120   NEXT
3129   REM trazar la linea hasta el centro para es te sector
3130   MOVE 0,0:DRAW x,y
3140 NEXT
3150 RETURN
```



Estos diagramas de sectores son más expresivos cuando los sectores se rellenan de color; la forma más sencilla de hacerlo es dibujar radios de colores. Para ello, se puede modificar el programa que teníamos al principio con las líneas que siguen:

```
3020 incremento=PI/500
3109 REM esta vez se dibujan lineas desde el centro
3110 MOVE centrox,centroy:DRAW centrox+radio*SIN(angulo),
centroy+radio*COS(angulo)
```



Ahora es incluso más lento, pero hay que darse cuenta de que, si no se juntan mucho los radios, hay puntos que quedan sin tocar y permanecen del color del fondo. Una idea para dar más velocidad sería colorear sólo uno de cada dos sectores, dejando los otros del color del fondo. Se ganara aún más tiempo si se elige cuidadosamente el orden de los sectores para que los que se rellenan sean los más pequeños. Entre otros defectos, esta solución tiene el inconveniente de

que obliga a rotular los sectores para que el diagrama no sea confuso.

Se puede aumentar la rapidez del programa teniendo almacenadas las coordenadas de los puntos de la circunferencia en un par de vectores, para no tener que calcularlos en el momento de realizar el diagrama. Si el programa está escaso de memoria es una mala solución, ya que se requieren dos vectores bastante largos. Ésta es la solución que adoptamos con el siguiente programa alternativo:

```
10 MODE 1
20 GOSUB 1000
30 GOSUB 2000
40 GOSUB 3000
50 END
999 REM muchas variables pueden ser enteras - esto aumenta la velocidad
1000 DEFINT a,c,n,r,v
1005 READ centrox,centroy
1010 READ radio
1020 READ numdeval
1030 DIM valor(numdeval),angulo(numdeval)
1039 REM sumar todos los valores para calcular las divisiones del circulo
1040 totalvalor=0
1050 FOR con=1 TO numdeval
1060   READ valor(con)
1070   totalvalor=totalvalor+valor(con)
1080 NEXT
1090 RETURN
1100 DATA 200,200,120,4,1,2,3,4
1998 REM radval influye en la velocidad del dibujo y en la densidad del
rellenado
1999 REM si se toma radio*3 mejora la velocidad aunque quedan puntos sin
rellenar
2000 radval=radio*10
2001 totangulo=0
2005 FOR con=1 TO numdeval
2010   angulo(con)=radval*valor(con)/totalvalor
2015   totangulo=totangulo+angulo(con)
2020 NEXT
2030 RETURN
3000 ORIGIN centrox,centroy
3010 DEFINT c
3020 contangulo=0
3030 color=1
3033 REM solo se necesita calcular un seno y un coseno por este metodo
3034 REM esto da mas velocidad
3035 elseno=SIN(2*PI/radval):elcoseno=COS(2*PI/radval)
3036 x1=radio:y1=0
```

```
3037 REM calculo de las coordenadas de los punto s de la circunferencia
3038 LOCATE 5,10:PRINT "espere, por favor":GOSUB 4000:CLS
3039 REM para modo 0 hacer numdecol=15 (sin cont ar el color del fondo)
3040 numdecol=3
3050 FOR con=1 TO numdeval
3069   REM color del sector
3070   color=1+(color+1)MOD numdecol
3079   REM asegurarse de que el primero y el ultim o sector son de colores
diferentes
3080   IF con=numdeval AND numdeval MOD numdecol=1 THEN color=1+(color+1)MOD
numdecol
3090   REM
3099   REM dibujo del sector
3100   FOR con1=contangulo TO contangulo+angulo(con)
3114     MOVE 0,0:DRAW x(con1),y(con1),color
3120   NEXT
3124   REM posicion inicial del siguiente sector
3125   contangulo=contangulo+angulo(con)
3140 NEXT
3150 RETURN
3997 REM normalmente convendra realizar este calculo en los programas largos
3998 REM en algun momento del programa en el que la espera
3999 REM no resulte excesivamente larga
4000 DIM x(totangulo),y(totangulo)
4010 FOR con=1 TO totangulo
4020   x=x1*elcoseno-y1*elseno
4030   y=x1*elseno+y1*elcoseno
4040   x(con)=x
4050   y(con)=y
4060   x1=x:y1=y
4070 NEXT
4080 RETURN
```

Observará que es difícil rellenar completamente una superficie con un color. La única forma de hacerlo bien es dibujar uno por uno los puntos individuales, como veremos en el capítulo siguiente.

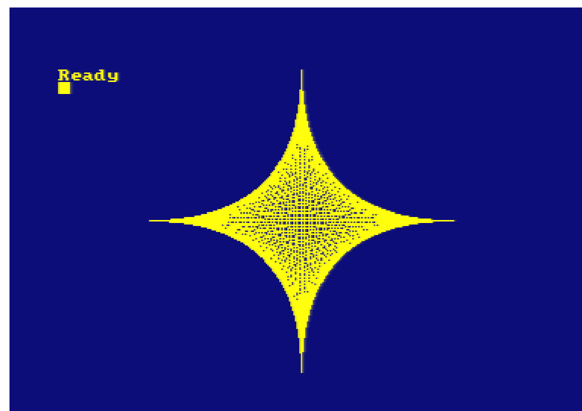
Ejercicios

1. Modifique el programa que dibuja diagramas de sectores de forma que además los rotule.
2. Modifique el programa para que dibuje en la pantalla varios diagramas de sectores con el mismo radio. (Aquí sí que es importante almacenar las coordenadas de la circunferencia en dos vectores, para no calcular varias veces los mismos datos.)
3. Escriba un programa que dibuje diagramas de sectores sucesivos, unos encima de otros, con los sectores rellenos de color.

4. Formas y dibujos

En el capítulo 1 hemos visto que es muy fácil realizar figuras geométricas con sólo los comandos MOVE y DRAW:

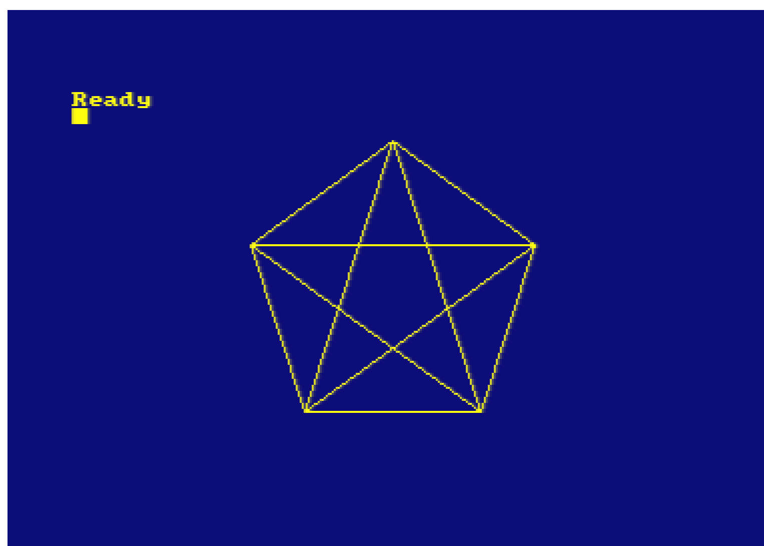
```
10 MODE 1
20 x=320:y=200
28 REM maximo da la longitud de los brazos de la figura
29 REM pruebe cambiando maximo y paso
30 maximo=200
40 paso=5
50 FOR con=0 TO maximo STEP paso
60   MOVE x-con,y
70   DRAW x,y+(maximo-con)
80   DRAW x+con,y
90   DRAW x,y-(maximo-con)
100  DRAW x-con,y
110 NEXT
```



En las clases de geometría para niños se hacen muchas figuras de este estilo. Se producen algunos efectos bastante espectaculares tomando una serie de puntos y uniéndolos 'todos con todos' mediante rectas. El siguiente programa construye un polígono con el número de lados que se desee y después une cada vértice con todos los demás:

```
10 MODE 1
20 radio=150
30 x=320:y=200
40 INPUT"Cuantos lados tiene la figura";lados
50 CLS
60 paso=2*PI/lados
70 DIM x(lados),y(lados)
80 con=0
90 ORIGIN x,y
```

```
100 MOVE 0,radio
110 FOR angulo=0 TO 2*PI STEP paso
120   DRAW radio*SIN(angulo),radio*COS(angulo)
130   x(con)=radio*SIN(angulo):y(con)=radio*COS(angulo)
140   con=con+1
150 NEXT
155 DRAW 0,radio
160 FOR con1=1 TO lados-1
170   FOR con2=con1+1 TO lados
180     MOVE x(con1),y(con1)
190     DRAW x(con2),y(con2)
200   NEXT
210 NEXT
```



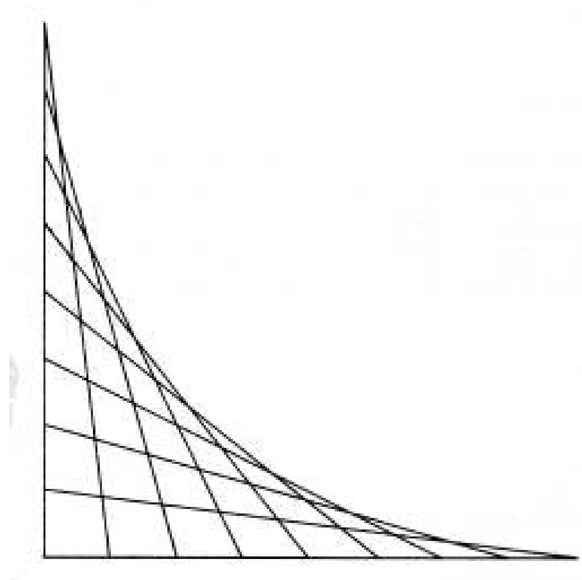
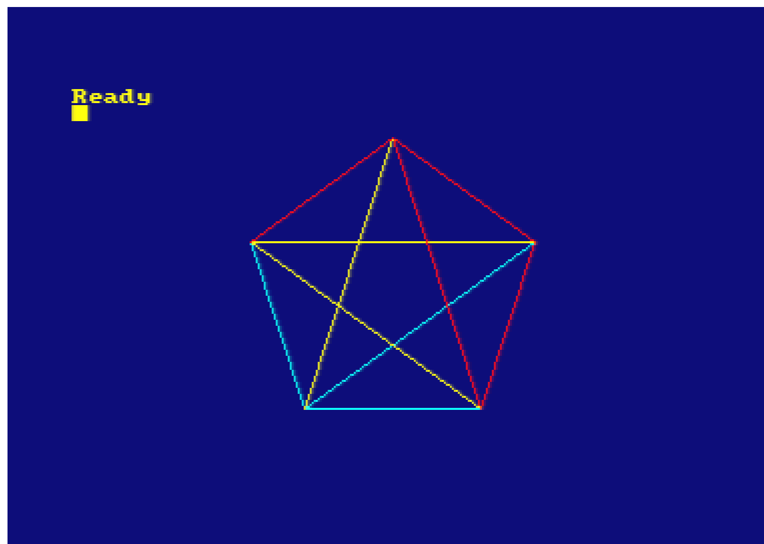


Figura 4.1. Ejemplo de curva formada uniendo puntos.

Queda mucho mejor si se dibuja con colores:

```
155 color=1:DRAW 0,radio,color
160 FOR con1=1 TO lados-1
170   FOR con2=con1+1 TO lados
173     REM experimente con los colores en la linea 175
174     REM cambie el MOD para ver el efecto
175     color=1+(color+1) MOD 3
180     MOVE x(con1),y(con1)
190     DRAW x(con2),y(con2),color
200   NEXT
210 NEXT
```



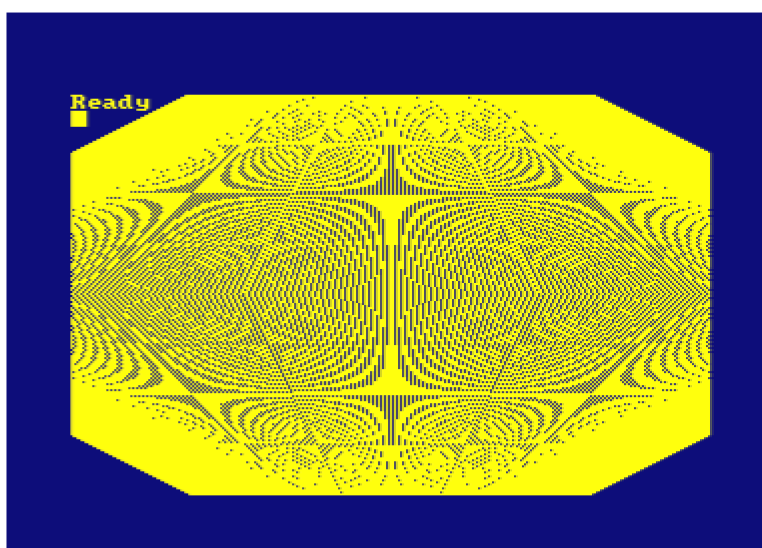
En este capítulo vamos a ver cómo se pueden producir figuras mucho mm elaboradas. Muchas de ellas están basadas en el uso de las funciones sen y coseno. Pero no se preocupe por las matemáticas; los programas están completamente escritos, aunque proporcionan muchas oportunidades para que usted dé otros valores a ciertas variables para estudiar lo que ocurre.

Tejidos de colores

Con el comando ORIGIN y con los desplazamientos relativos es fácil dibujar figuras con simetría. El siguiente programa coloca el origen en centro de la pantalla y crea una figura uniendo puntos del nuevo eje con puntos de la parte superior y de la parte inferior de la pantalla. La coordenada X se multiplica por un factor que hace que las líneas converjan o diverjan:

```
10 MODE 1
19 REM se usan numeras enteros para mayor velocidad
20 DEFINT c,f,x,y
30 xorigen=320:yorigen=200
39 REM cambie estos factores para obtener diferentes efectos
```

```
40 factor1=3:factor2=1
50 factord=factor1-factor2
60 ORIGIN xorigen,yorigen
69 REM el bucle dibuja cuatro rectas de posicion variable
70 FOR con=0 TO 200
80    MOVE 0,0:MOVER con*factor1,0
90    DRAWR -con*factord,200
100   MOVER -con*factor2*2,0
110   DRAWR -con*factord,-200
120   DRAWR con*factord,-200
130   MOVER con*factor2*2,0
140   DRAWR con*factord,200
150 NEXT
```

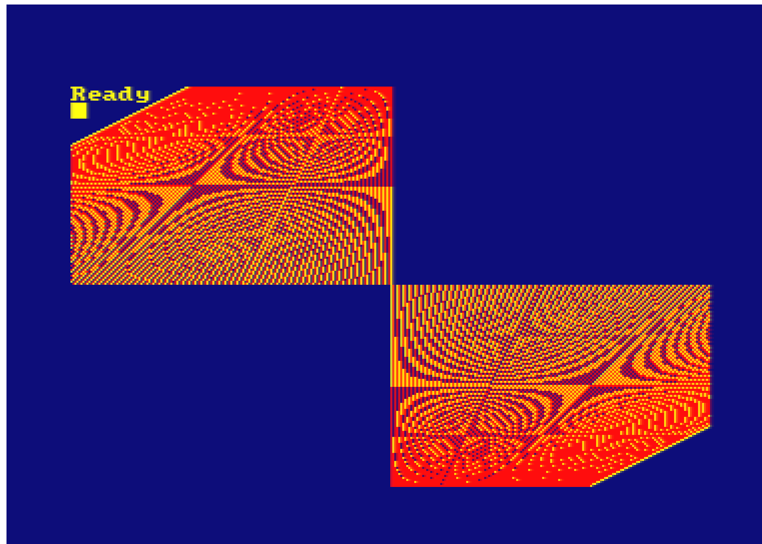


Observe que el DEFINT de la línea 20 está puesto para dar rapidez al programa. No siempre será posible trabajar con variables enteras a causa de los senos y cosenos, que dan siempre decimales.

Los factores de la línea 40 se pueden cambiar, pero conviene evitar que sean mayores de 15, pues entonces las líneas se separan excesivamente. Las sorprendentes formas que surgen en el dibujo provienen de que ciertos puntos quedan del color del fondo y también de los puntos en que las líneas se superponen. Experimente con factores diferentes y ejecute el programa en otros modos de pantalla. Obtendrá una bonita alfombra añadiendo estas líneas:

```
35 color1=1:color2=1
70 FOR con=0 TO 200
80 MOVE 0,0:MOVER con*factor1,0
90 DRAWR -con*factord,200,color1
100 MOVER -con*factor2*2,0
110 DRAWR -con*factord,-200,color2
120 DRAWR con*factord,-200,color1
130 MOVER con*factor2*2,0
```

```
140 DRAWR con*factord, 200,color2
150 color1=1+(color1+1) MOD 4: color2=1+(color2+1) MOD 4
160 NEXT
```



La línea 150 hace que las rectas se dibujen usando cíclicamente los colores del modo 1. La sentencia MOD sirve para dar el resto de la división; por ejemplo, 5 MOD 4 es 1. Además, se suma un 1 para garantizar que el resultado no da el color del fondo; de otra manera, como 4 MOD 4=0 se obtendrían rectas que no contrastarían con el fondo. Las líneas 35 y 150 hacen que el aspecto final sea de un cálido color naranja. Las tintas que se usan en las sentencias DRAWR son 1 (amarilla) y 3 (roja). De lo apretadas que resulten las líneas (lo que depende de los factores de la línea 40) depende que el color quede naranja o a rayas rojas y amarillas.

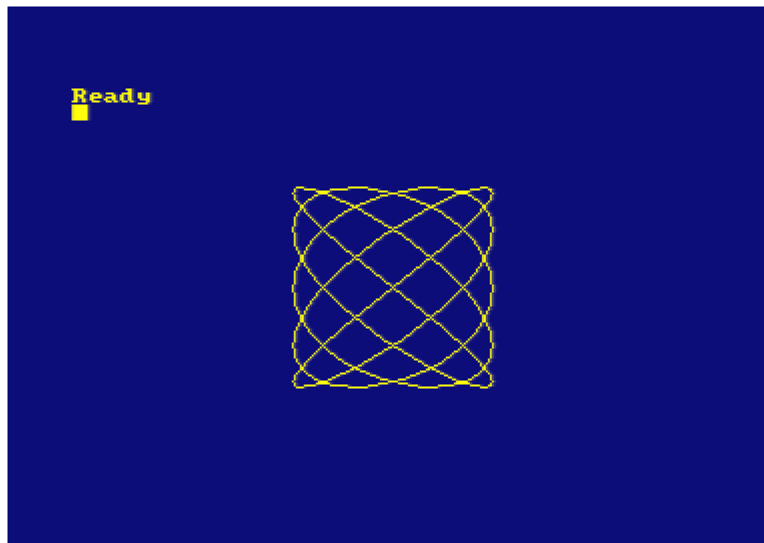
Ajustando los valores de colorí y color2 y cambiando MOD 4 en la línea 150 por MOD 2 o MOD 3, se obtiene cierta variedad de efectos: figuras con cuadrantes opuestos en diferentes colores o compuestos únicamente de ciertos colores. Al ejecutar el programa en los otros modos cambian completamente los resultados, salvo que se ajusten los colores utilizados.

Figuras de Lissajous

Las figuras de Lissajous se obtienen utilizando una idea de la que ya nos servimos para dibujar una circunferencia. Entonces manteníamos constante el radio y tomábamos el seno y el coseno del mismo ángulo para construir un punto de la circunferencia. Variando este ángulo se obtienen figuras diversas:

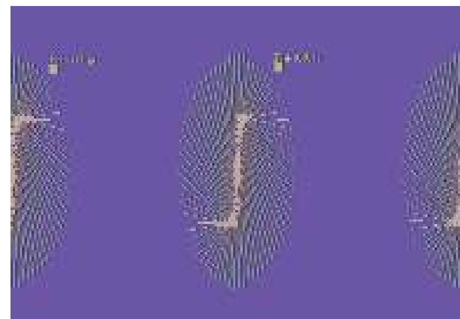
```
10 MODE 1
20 xorigen=320:yorigen=200
30 ORIGIN xorigen,yorigen
40 MOVER 100,0
50 FOR angulo=0 TO 32 STEP PI/30
60   DRAW 100*COS(angulo),100*SIN(angulo*0.8)
```

```
70 NEXT
```



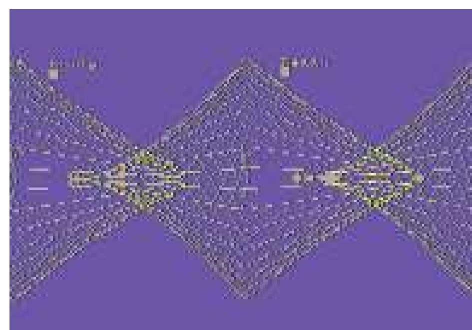
Una alternativa puede ser unir con rectas los puntos de dos de estas curvas:

```
10 MODE 1
20 xorigen=320:yorigen=200
30 ORIGIN xorigen,yorigen
40 MOVER 100,0
50 FOR angulo=0 TO 6.4 STEP PI/35
55   MOVE 200*SIN(angulo),100*COS(angulo)
60   DRAW 100*COS(angulo),200*SIN(angulo)
70 NEXT
```



O bien este otro ejemplo:

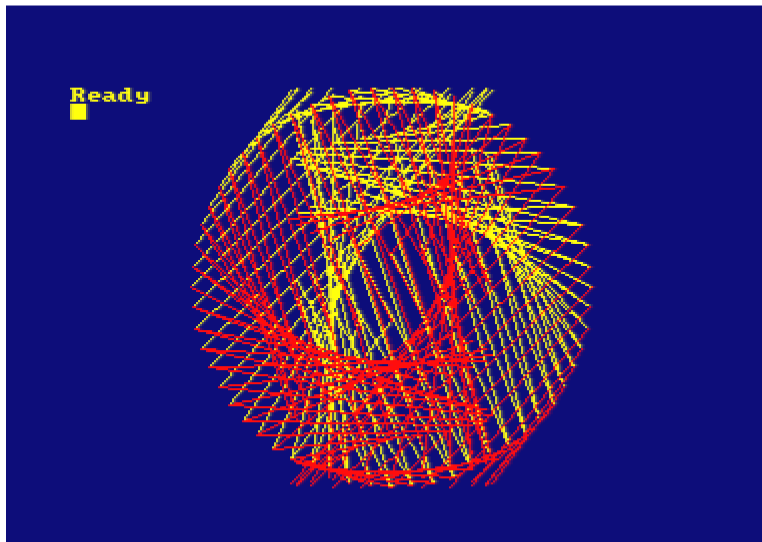
```
10 MODE 1
20 xorigen=320:yorigen=200
30 ORIGIN xorigen,yorigen
50 FOR angulo=0 TO 6.4 STEP PI/35
55   MOVE 300*SIN(angulo),50*COS(angulo)
60   DRAW 10*COS(angulo/5),200*SIN(angulo*2)
70 NEXT
```



O también con color:

```
10 MODE 1
20 xorigen=320:yorigen=200
30 ORIGIN xorigen,yorigen
40 color=1
50 FOR angulo=0 TO 20 STEP PI/30
```

```
53   IF angulo>10 THEN color=3
55   MOVE 200*SIN(angulo),200*COS(angulo)
60   DRAW 100*COS(angulo*3),200*SIN(angulo/3),color
70 NEXT
```



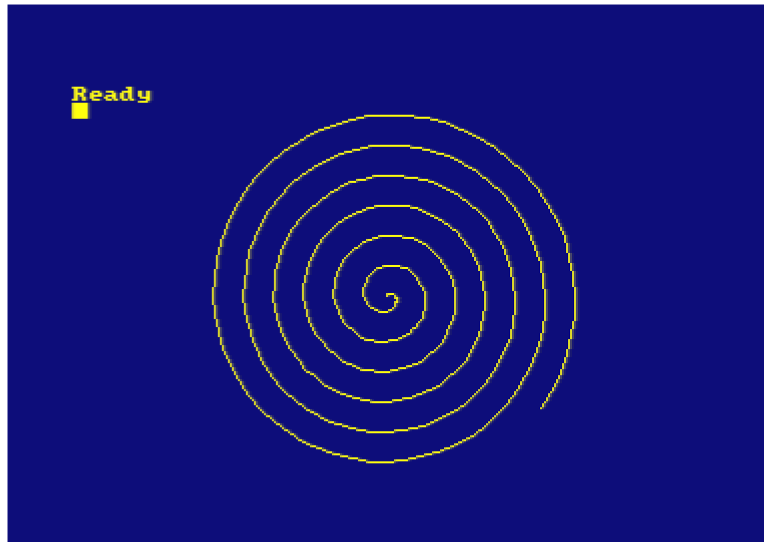
Como en otras ocasiones, puede ensayar utilizando otros modos. El empleo del seno y del coseno para producir este tipo de dibujos son la consecuencia de bastantes experimentos, aunque al principio los resultados parezcan impredecibles. Sin embargo, pronto se aficionará a experimentar con el cambio de las variables o a investigar con otras funciones, como los cuadrados de senos y cosenos o sus productos por otros factores.

Espirales

Podemos crear una espiral utilizando nuestro programa de dibujar circunferencias pero haciendo que el radio cambie constantemente:

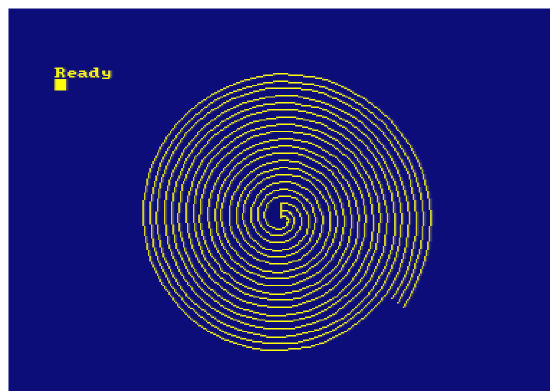
```
10 MODE 1
20 GOSUB 1000
100 END
1000 xorigen=315:yorigen=190
1010 ORIGIN xorigen,yorigen
1020 color=1
1029 REM valor del incremento del radio de la espiral cada vez que se dibuja un nuevo punto
1030 radioincremento=0.5
1040 paso=PI/30
1049 REM angulofinal determina las vueltas que dara la espiral
1049 REM un valor excesivo hace que se salga de la pantalla
1050 angulofinal=40
1059 REM se comienza con radio 1
```

```
1060 radiocom=1
1070 GOSUB 2000
1900 RETURN
2000 MOVE 0,0
2010 FOR angulo=0 TO angulofinal STEP paso
2020   DRAW radiocom*SIN(angulo),radiocom*COS(angulo),color
2030   radiocom=radiocom+radioincremento
2040 NEXT
2050 RETURN
```



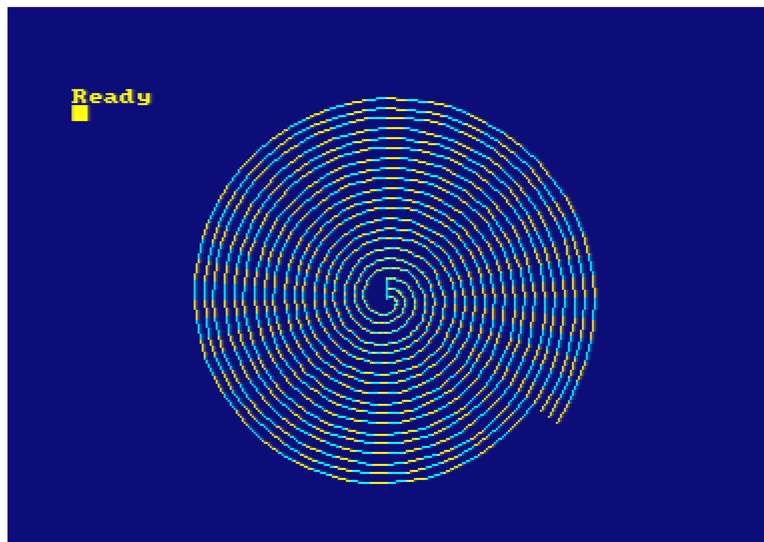
Añadiendo algunas líneas podemos hacer que se dibujen unas espirales dentro de otras:

```
1079 REM otra espiral en el interior de la anterior; esta comienza con radio 10
1080 radiocom=10
1090 GOSUB 2000
1099 REM ultima espiral en el interior de la anterior; esta comienza con radio
20
1100 radiocom=20
1110 GOSUB 2000
```



La espiral puede animarse de un aparente movimiento de rotación dibujando los puntos de diferentes colores y poniendo tintas intermitentes entre un color y el del fondo:

```
10 MODE 1
20 GOSUB 1000
28 REM tintas que parpadeen entre dos colores
29 REM con alternancia entre ambas
30 INK 1,1,20
12 INK 2,20,1
50 respuesta$=""
60 WHILE respuesta$=""
70 respuesta$=INKEY$
80 WEND
90 INK 1,24:INK 2,20
100 END
1053 REM ahora las dos tintas son iguales
1054 REM al dar el mismo color a la tinta 1 que a la tinta 2
1055 INK 1,20
2014 REM las lineas se dibujan alternando las dos tintas
2015 IF color=1 THEN color=2 ELSE color=1
```



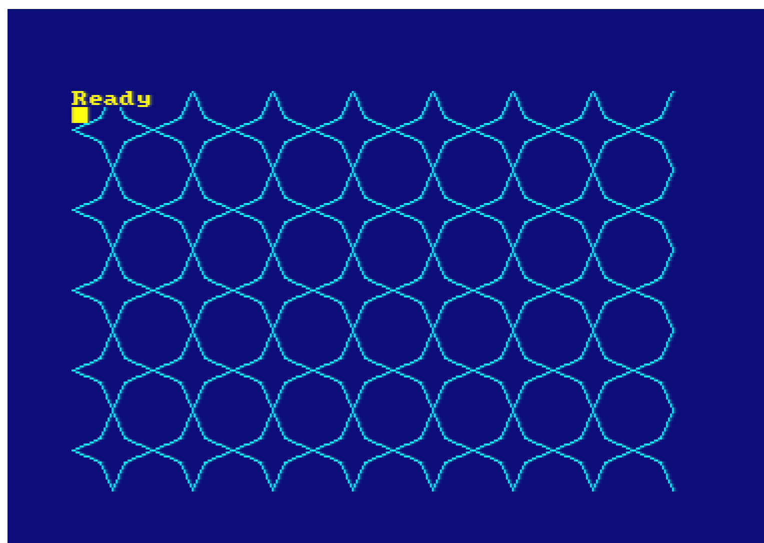
Es una técnica usual que volveremos a ver.

Figuras repetitivas

Muchos dibujos, como los habituales en los papeles pintados para las paredes, se obtienen mediante la simple repetición de una figura:

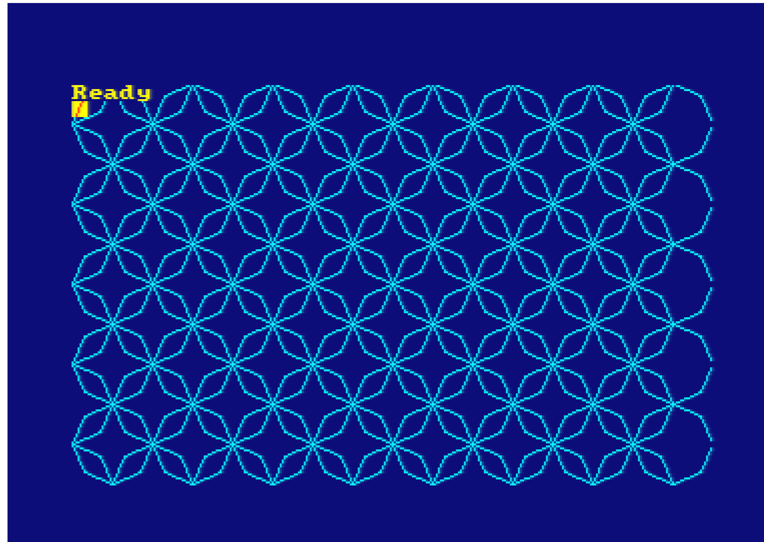
```
10 MODE 1
```

```
20 GOSUB 1000
40 END
998 REM datos para un octogono
999 REM pruebe con otras figuras
1000 xcom=0:ycom=0
1001 REM xcom,ycom forman el centro del primer poligono
1010 radio=40
1020 lados=8
1030 paso=2*PI/lados
1040 color=2
1050 GOSUB 3000
1060 RETURN
2998 REM se rellena la pantalla con copias del poligono
2999 REM se detiene cuando las coordenadas del centro se salen de la pantalla
3000 centrox=xcom:centroy=ycom
3010 WHILE centrox<639 OR centroy<399
3020   ORIGIN centrox,centroy
3030   MOVE 0,radio
3040   FOR angulo=0 TO 2*PI STEP paso
3050     DRAW radio*SIN(angulo),radio*COS(angulo),color
3060   NEXT
3069   REM desplazamiento en la direccion de la x
3070   centrox=centrox+radio*2
3079   REM si se sale de la pantalla se recomienza mas arriba
3080   IF centrox>639 AND centroy<399 THEN centrox=xcom:centroy=centroy+radio*2
3090 WEND
3100 RETURN
```



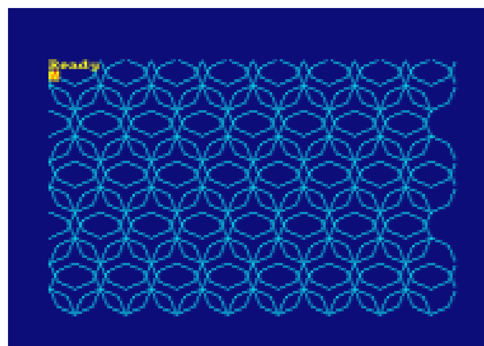
Se logran resultados más elaborados superponiendo un segundo conjunto de figuras al primero. Esto se puede hacer con figuras distintas o con figuras iguales pero de diferente tamaño:

```
30 GOSUB 2000
1999 REM datos para un hexagono
2000 xcom=40:ycom=40
2010 radio=40
2020 lados=6
2030 paso=2*PI/lados
2040 color=2
2030 GOSUB 3000
2060 RETURN
```



Las filas se pueden poner escalonadas comenzándolas en una de dos posiciones alternativas:

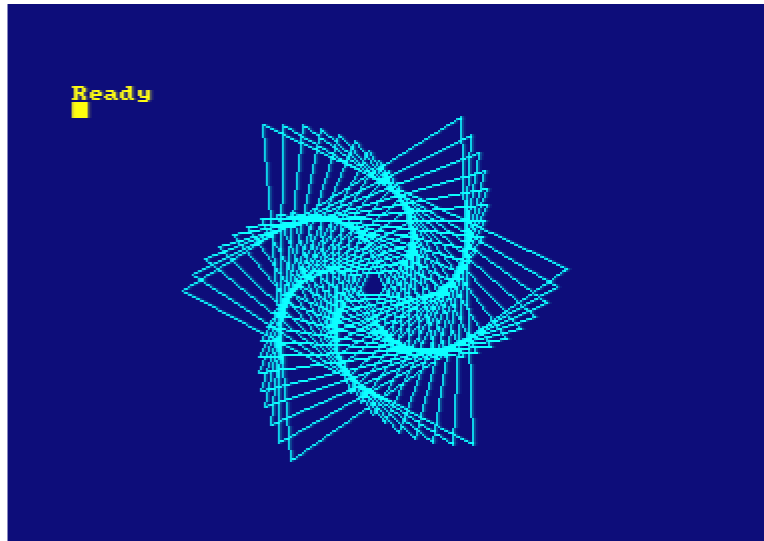
```
3079 REM si se sale de la pantalla se recomienza mas arriba
3080 IF centrox>639 AND centroy<399 THEN GOSUB 4000
3090 WEND
3100 RETURN
3999 REM se escalona el comienzo de cada fila de poligonos
4000 IF xcom=0 THEN xcom=radio ELSE xcom=0
4010 centrox=xcom:centroy=centroy+radio*2
4020 RETURN
```



Rotación de una figura

Es fácil modificar el programa de dibujar circunferencias para que dibuje polígonos. El programa así construido puede ser incorporado como subrutina en un programa que vaya alargando y rotando una figura, de manera que proporcione un dibujo en espiral:

```
10 MODE 1
20 GOSUB 1000
30 GOSUB 1500
40 END
999 REM lectura de los datos del poligono
1000 READ lados
1010 READ radio
1020 READ centrox,centroy
1030 READ radiocambio,angulocambio
1040 color=2
1050 paso=2*PI/lados
1060 angulocom=0:angulofin=2*PI
1070 RETURN
1500 ORIGIN centrox,centroy
1508 REM no se dibuja el poligono cuando se hace demasiado grande
1509 REM lo que significa radio>200 para los poligonos de los DATA
1510 WHILE radio<200
1520   MOVE radio*SIN(angulocom),radio*COS(angulocom)
1530   FOR angulo=angulocom TO angulofin STEP paso
1540     DRAW radio*SIN(angulo),radio*COS(angulo),color
1550   NEXT
1560   DRAW radio*SIN(angulocom),radio*COS(angulocom)
1569   REM se incrementa el radio
1570   radio=radio+radiocambio
1579   REM se gira el siguiente poligono para que forme angulo con el anterior
1580   angulocom=angulocom+angulocambio
1590   angulofin=angulofin+angulocambio
1600 WEND
1610 RETURN
2000 DATA 3,20,300,200,5,1
2010 DATA 3,20,300,200,3,10
2020 DATA 4,30,300,200,4,3
2030 DATA 6,20,300,200,1,6
2040 DATA 6,20,300,200,6,1
2050 DATA 8,10,300,200,5,10
2060 DATA 8,10,300,200,5,2
```



Ejecute el programa tal como está. Los DATA de la línea 2000 y siguientes proporcionan datos para varias figuras diferentes. Borre luego el primer DATA y vuelva a ejecutarlo, y así repetidamente hasta agotar los ejemplos. De esta forma irá viendo el efecto que produce el cambio de número de lados del polígono y de los coeficientes de alargamiento y de rotación.

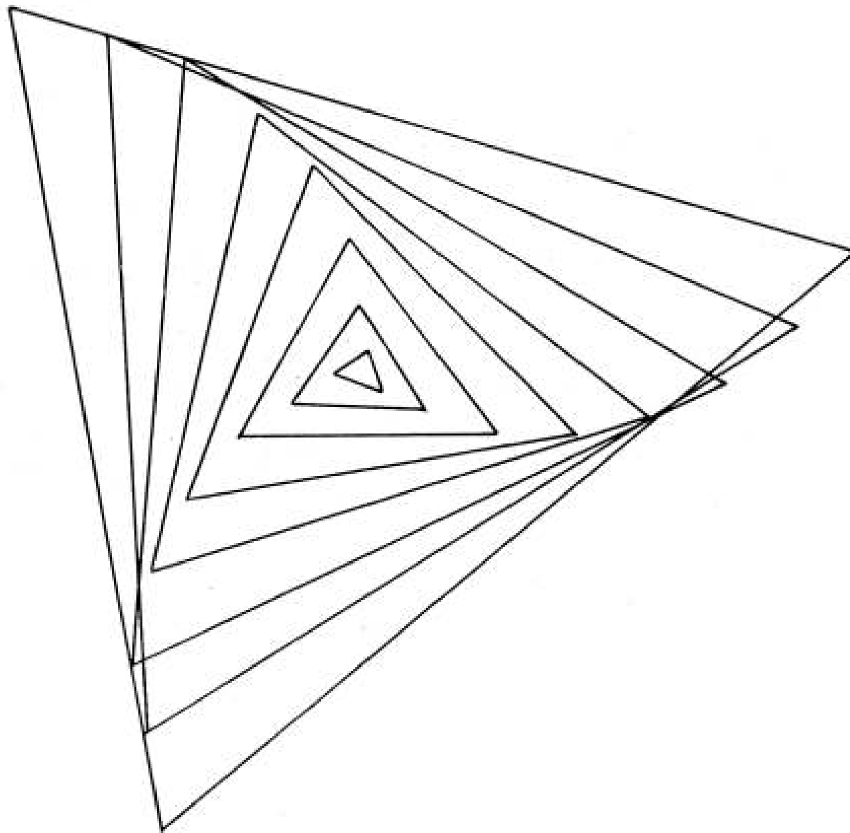
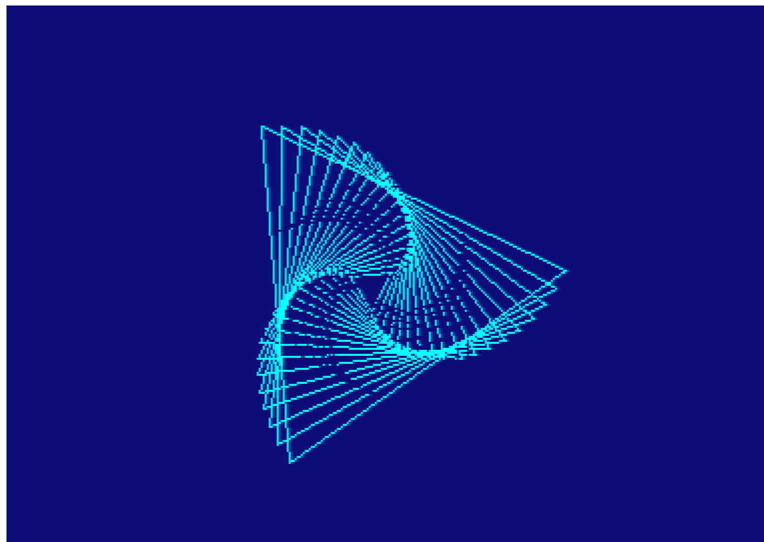


Figura 4.2. Tipo de figura que se obtiene por rotación de una forma básica

Con el uso juicioso del color se pueden lograr efectos llamativos:

```
10 MODE 1
20 GOSUB 1000
29 REM las tintas 2 y 3 sol intermitentes y alternadas entre si
30 GOSUB 1500
40 INK 2,1,20
50 INK 3,20,1
59 REM se espera la pulsacion de una tecla
60 respuesta$=""
70 WHILE respuesta$=""
80 respuesta$=INKEY$
90 WEND
99 REM tintas normales
100 INK 2,20
110 INK 3,6
120 END
1504 REM tinta 3 igual a la 2
1505 INK 3,20
1594 REM los poligonos se dibujan alternativamente con una y otra tinta
1595 IF color=2 THEN color=3 ELSE color=2
```



Al terminar de dibujar la figura, el programa produce un sorprendente efecto por la intermitencia entre el color del fondo y el otro color.

Ejercicios

1. 1.Escriba una nueva versión del programa que une los vértices de un polígono de manera que dibuje dos polígonos y una sus vértices.
2. Investigue el efecto que producen en el programa de los tejidos de colores los cambios siguientes: otro paso distinto en el bucle FOR..NEXT; dibujar sobre la primera figura otra con factores y colores diferentes; desplazar el

origen para crear una superficie llena de las mismas figuras.

3. Modifique el programa de rotación de polígonos para que el número de lados del polígono varíe a lo largo del programa. Vaya alternando, por ejemplo, entre un triángulo y un pentágono; o bien entre todos los polígonos, aumentando un lado en cada rotación hasta que haya 20 lados y recomenzando de nuevo.

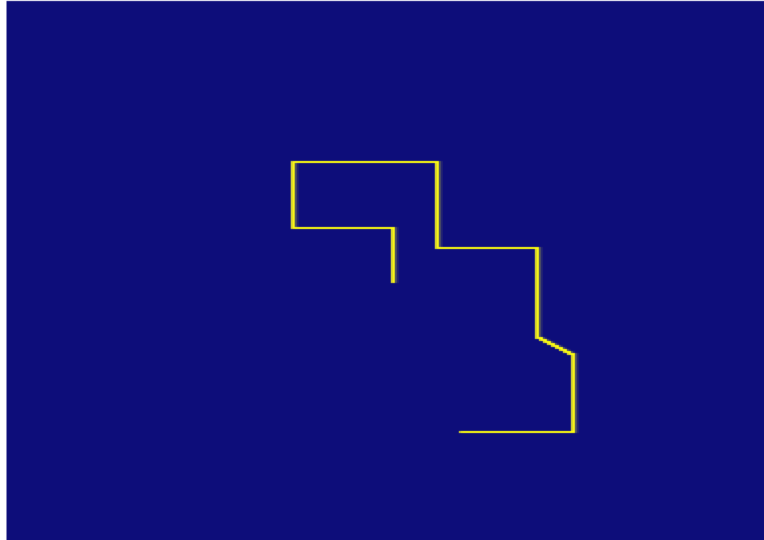
DIBUJO INTERACTIVO EN LA PANTALLA

Además de hacer que el ordenador construya figuras más o menos geométricas, también se puede conseguir que ayude al usuario para que sea él mismo el que diseñe una figura sobre la pantalla; a esto vamos a dedicar el resto del capítulo. Para que los programas puedan ser utilizados por la totalidad de los usuarios, los hemos diseñado para que funcionen empleando el teclado, pero es fácil hacer las pequeñas modificaciones que se requieren para utilizar los mandos de juego ("joysticks"). Utilizaremos el modo 0 de pantalla para proporcionar mayor variedad de colores.

El primero de los programas proporciona la herramienta más elemental para dibujar sobre la pantalla. Permite dibujar puntos que vayan formando líneas de la dirección deseada:

```
10 MODE 0
20 x=320:y=200
30 color=1
40 MOVE x,y
50 PLOT x,y,color
60 GOSUB 1000
70 END
997 REM examen del teclado - fin cuando se pulsa la tecla 'e'
1000 WHILE respuesta$<>"e"
1010 respuesta$=INKEY$
1019 REM comandos para dibujar la linea: arriba/abajo=a/z  izquierda/derecha=,/.
1020 IF respuesta$="a" THEN y=y+2
1030 IF respuesta$="z" THEN y=y-2
1040 IF respuesta$="," THEN x=x-4
1050 IF respuesta$="." THEN x=x+4
1060 PLOT x,y,color
1070 WEND
1080 RETURN
```

El primer punto se dibuja en las coordenadas de la línea 20. Luego, el programa espera que se pulse una tecla para avanzar hasta otro punto. Las direcciones de avance se controlan con las teclas siguientes: 'a' para arriba, 'z' para abajo, para izquierda y para derecha. El programa actualiza las coordenadas en función de la tecla pulsada y dibuja el punto.



Es fácil añadir otros comandos a esta estructura básica. Aunque el movimiento está por el momento limitado a las cuatro direcciones cardinales, se pueden programar avances en diagonal añadiendo líneas a partir de la 1050.

Tal como está, el programa dibuja líneas continuas, y no es posible pasar de una posición a otra separada de ella sin trazar una línea. Podemos añadir como opción que el color del punto pueda ser tanto el de dibujo como el del fondo:

```
1051 IF respuesta$="d" THEN color=1
1052 IF respuesta$="f" THEN color=0
```

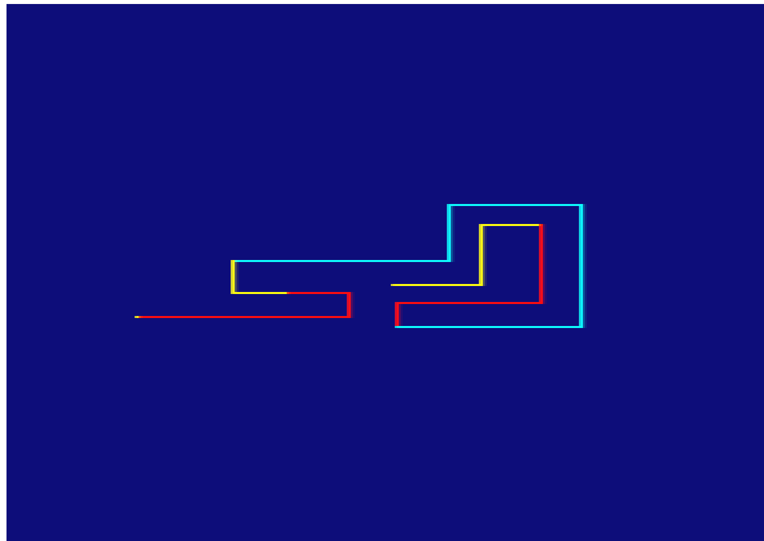
Pero si se ha seleccionado el color del fondo no se verá el dibujo de los puntos y no sabremos dónde estamos dibujando. Esto puede arreglarse dibujando dos veces el punto; la primera en un color que se vea, y la segunda en su color definitivo:

```
35 colorvisible=1
1005 PLOT x,y,color
1060 PLOT x,y,colorvisible
```

Si el color que se selecciona es el del fondo, el punto quedará intermitente entre los dos colores, actuando como un cursor que nos indica por dónde vamos.

También se puede dar la opción de seleccionar el color del dibujo, bien directamente con varias teclas, bien empleando una sóla tecla para recorrer un ciclo de colores de la misma forma que hicimos en programas anteriores:

```
1053 IF respuesta$="c" THEN color=1+(color+1) MOD 3
```



De esta manera, tenemos una elección entre tres colores que se distinguen perfectamente del color del fondo.

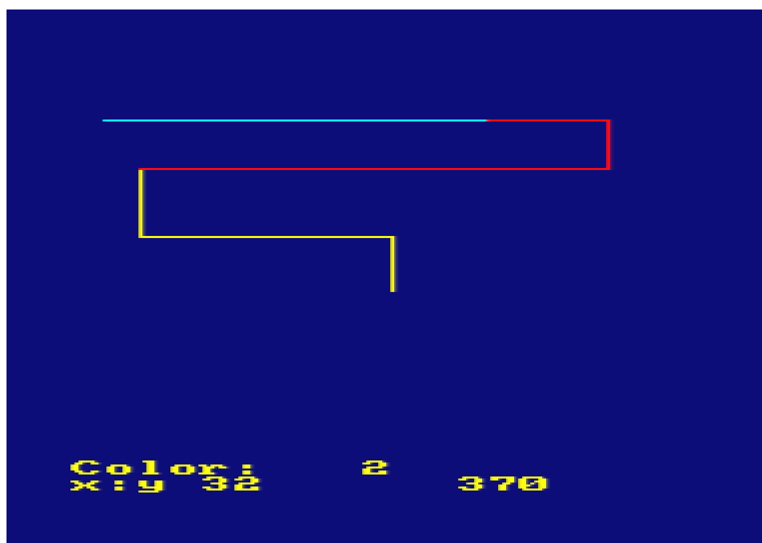
A veces, será difícil dibujar con exactitud rectángulos y otras figuras parecidas empleando la vista como único sistema de medir distancias. Podemos hacer que nos vaya apareciendo una información numérica en la pantalla. Esto nos da una disculpa para presentar el comando WINDOW (*window* es ventana), que podemos emplear para hacer que todo lo que sea texto aparezca en una zona determinada de la pantalla:

```
15 WINDOW 1,40,24,25
16 PRINT "Color: "
17 PRINT "x:      y: "
```

Los cuatro números que siguen a WINDOW especifican primero la coordenada X de los bordes izquierdo y derecho de la ventana, y luego la coordenada Y de los bordes superior e inferior. En nuestro caso, la ventana para el texto ocupará las dos últimas líneas de la pantalla:

```
1051 IF respuesta$="d" THEN color=colorvisible
1052 IF respuesta$="f" THEN colorvisible=color:color=0
1053 IF respuesta$="c" THEN color=1+(color+1) MOD 3
1060 PLOT x,y,colorvisible
1065 GOSUB 2000
1070 WEND
1080 RETURN
2000 IF colorviejo<>color THEN LOCATE 9,1:PRINT color
2010 IF xviejo<>x THEN LOCATE 4,2:PRINT x;
2020 IF yviejo<>y THEN LOCATE 12,2:PRINT y;
2030 colorviejo=color
2040 xviejo=x:yviejo=y
2050 RETURN
```

Ahora podremos realizar nuestros cálculos con mayor exactitud, pues el programa nos da una información que mantiene siempre actualizada sobre el color y las coordenadas del punto. Observe que el Amstrad considera también la ventana de texto como parte de la pantalla gráfica; lo podrá comprobar llevando su dibujo hacia esa zona.



Hay aún muchos defectos en el programa. Las líneas sólo se pueden borrar dibujando sobre ellas con el color del fondo. Por otra parte, si se cruza sobre una línea al ir dibujando con el color del fondo, se borrará el punto de intersección. A pesar de que esto pueda parecer un problema sin solución, tiene una muy sencilla, aunque para ponerla en práctica vamos a tener que utilizar nuestros conocimientos sobre los números binarios.

Ejercicios

1. El programa anterior no se preocupa de si los puntos se salen de la pantalla. Haga una modificación para que el programa impida salirse de la pantalla o dibujar sobre la ventana de texto.
2. Aumente la posibilidad de elegir colores hasta 16. Utilice el comando INK para elegir los 16 que más convengan de entre los 27 posibles, en lugar de utilizar los que vienen dados por defecto.
3. Añada los comandos necesarios para que se pueda dibujar en diagonal.

La opción XOR

Ahora podremos comprobar el interés de los comentarios que hicimos en el primer capítulo acerca de los sistemas binario y hexadecimal, ya que son conceptos importantes cuando se trata de las operaciones gráficas.

Conviene recordar que la pantalla es la representación externa de una parte de

la memoria del ordenador. Los caracteres que aparecen o las líneas que están dibujadas son el resultado de la forma en que se almacenan valores binarios en dicha zona de memoria, que es la que el Amstrad examina para construir la configuración de la pantalla.

El dibujo de una línea hace que los bytes (los grupos de 8 bits) cambien en la parte de memoria que corresponda. El valor de una posición de memoria es lo que determina si un punto de la pantalla está encendido o apagado y, si está encendido, de qué color lo está. De hecho, el color de cada punto se almacena en un determinado byte, en una forma que no es obvia pero que tampoco nos interesa por el momento. Para dar una explicación sencilla, vamos a imaginarnos que hay una correspondencia uno a uno entre puntos y bytes, de manera que la configuración de los bits de un byte es lo que indica al Amstrad cómo debe dibujar el correspondiente punto en la pantalla. Esto no es exactamente así, pero el modelo servirá perfectamente para lo que necesitamos comprender.

Supondremos que se puede elegir entre cuatro colores y que el byte que representa un punto de pantalla puede tener uno de los cuatro valores de la figura 4.3.

Cuando la pantalla está completamente azul, todos los bytes valen 00000000; si está completamente blanca, todos valen 00000001 y así sucesivamente. Al dibujar una línea blanca, los bytes correspondientes a los puntos de la línea toman el valor 00000001; lo mismo, pero con el valor 00000010, cuando se dibuja una línea amarilla.

CÓDIGO BINARIO	COLOR
00000000	Azul
00000001	Blanco
00000010	Amarillo
00000011	Rojo

Figura 4.3. Representación binaria de los colores en el supuesto ficticio.

Dijimos hace tiempo que había códigos ASCII que representaban comandos del Amstrad como, por ejemplo, movimientos del cursor o borra do de la pantalla. Uno de estos códigos establece la modalidad en qui el Amstrad dibuja los puntos.

Se puede hacer que el Amstrad dibuje en la modalidad de 'or exclusivo' (que se abrevia por 'EXOR' o, más frecuentemente, por 'XOR').

Cuando no se emplea esta opción, el Amstrad se limita a cambiar el anterior valor del byte por el nuevo. Por ejemplo, al dibujar una línea amarilla, los bytes correspondientes tomaran el valor 00000010.

Con la opción XOR el Amstrad combina el viejo valor del byte con el nuevo, según ciertas reglas. Vamos a examinar lo que ocurriría con un byte individual

correspondiente a un punto de la pantalla, para ver que el efecto es el que nos interesa. Supongamos para empezar que el byte tiene el valor 00000000, o sea, representa un punto del color azul del fondo, como en la figura 4.4:

00000000	Un punto del color del fondo (azul)

Figura 4.4.

Supongamos que se dibuja una línea amarilla (valor 00000010) sobre dicho punto. La situación es la que indica la figura 4.5:

00000000	Un punto azul
00000010	Y una línea amarilla sobre el punto

Figura 4.5.

Como el Amstrad está empleando la opción XOR, lo que hace no e reemplazar 00000000 (azul) por 00000010 (amarillo), sino mezclar los do bytes. Donde los bits de ambos bytes son diferentes, pone un 1 y, donde son iguales, pone un 0. El resultado es que el valor que queda corresponde a un punto amarillo, como se puede ver en la figura 4.6. O sea, ocurre lo mismo de siempre. Desde luego esto no es lo que el lector podría esperar después de la publicidad que hemos hecho de la opción XOR.

00000000	Un punto azul
XOR 00000010	cruzado por una línea amarilla
00000010	da un punto amarillo

Figura 4.6.

Veamos ahora qué ocurre con esta opción si se dibuja una línea amarilla cruzando otra línea ya dibujada. El efecto, que se muestra en la figura 4.7, es más bien inesperado. Como coinciden los bytes antiguo y nuevo, la opción XOR produce el valor 00000000, correspondiente al color azul del fondo; la línea desaparece de la misma forma que si se hubiese dibujado en azul con la opción normal.

00000010	Un punto amarillo
XOR 00000010	cruzado por una línea amarilla
00000000	da un punto de color del fondo (azul)

Figura 4.7.

Si a continuación se dibuja encima otra línea amarilla, la línea amarilla original reaparece, puesto que se produce de nuevo la situación de la figura 4.6.

¿Qué ocurre cuando se dibuja (con la opción XOR) una línea amarilla sobre una línea blanca?

00000001	Un punto blanco
XOR 00000010	cruzado por una línea amarilla

Figura 4.8.

Resulta una línea roja, como explica la figura 4.9:

00000001	Un punto blanco
XOR 00000010	cruzado por una línea amarilla
00000011	da un punto rojo

Figura 4.9.

00000011	Un punto rojo
XOR 00000011	cruzado por una línea rojo
00000000	da un punto de color del fondo (azul)

Figura 4.10.

Es posible que la opción XOR no le parezca muy útil, pero los efectos gráficos que consigue son de un valor inestimable. Permite dibujar líneas o borrarlas sin afectar a las demás; así se puede ensayar con un dibujo y corregirlo, antes de

hacerlo definitivo. Todo lo que hace falta recordar es que, con la opción XOR, lo que se dibuja dos veces desaparece.

Dibujo con la opción XOR

Antes de comenzar el diseño de un nuevo programa de dibujo que aproveche las posibilidades de la opción XOR, vamos a ver qué facetas podríamos tratar de incluir:

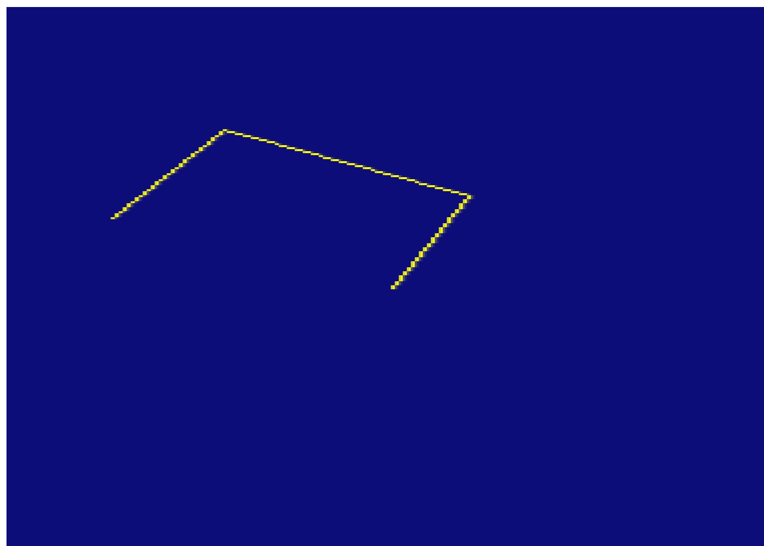
1. que las líneas sean provisionales o permanentes;
2. que se puedan borrar líneas;
3. que el dibujo se pueda grabar en cinta;
4. dibujo automático de figuras predefinidas, como circunferencias o rectángulos;
5. traslación, ampliación a escala y otros tipos de movimientos de las figuras.

De las dos últimas funciones ya trataremos en un capítulo posterior. Para incluir 1) no hay problema después de lo que acabamos de ver. Para realizar 2) y 3), lo mejor sería tener almacenadas en una matriz las coordenadas de los extremos de las rectas. Teniendo estos datos almacenados es sencillo identificar una línea y borrarla. También sería fácil almacenar el dibujo en un fichero; bastaría con guardar esta matriz de coordenadas, que se podría utilizar para reproducir el dibujo en cualquier momento.

Vamos a realizar un programa de diseño modular, para poder ampliarlo fácilmente:

```
10 MODE 0
20 DIM x(100),y(100)
30 xant=320:yant=200
40 x=320:y=200
50 colorvisible=1
60 PRINT CHR$(23);CHR$(1);
70 GOSUB 1000
80 GOSUB 2000
90 END
999 REM dibujo de la recta
1000 PLOT x,y,colorvisible
1010 DRAW xant,yant
1020 RETURN
2000 WHILE respuesta$<>"e"
2010   GOSUB 1000
2020   respuesta$=LOWER$(INKEY$)
2030   IF respuesta$="a" THEN y=y+2
2040   IF respuesta$="z" THEN y=y-2
2050   IF respuesta$="," THEN x=x-4
2060   IF respuesta$="." THEN x=x+4
```

```
2070   IF respuesta$=" " THEN GOSUB 3000
2080   GOSUB 1000
2090 WEND
2100 RETURN
2999 REM opcion grafica normal para dibujar de m anera permanente
3000 PRINT CHR$(23);CHR$(0);
3010 GOSUB 1000
3019 REM vuelta a la opcion XOR
3020 PRINT CHR$(23);CHR$(1);
3030 con=con+1
3040 x(con)=x:y(con)=y
3050 xant=x:yant=y
3060 RETURN
```



La línea 20 reserva matrices para las coordenadas de 100 puntos (esta cantidad puede perfectamente aumentarse). Las variables x e y son las coordenadas del punto con el que se está trabajando en ese momento. Las variables *xant* e *yant* son las coordenadas del último punto que se ha fijado definitivamente. Éstas son las coordenadas que deben estar disponibles para poder dibujar si se desea una recta permanente entre ambos puntos.

La subrutina 2000 es el módulo central del programa. Se encarga de consultar el teclado y dibuja y borra sucesivamente una recta entre (x,y) y (*xant*,*yant*).

La subrutina 1000 se encarga de dibujar la recta entre ambos puntos cuando uno de ellos va desplazándose; es una técnica que podría llamarse de la 'cinta elástica' por el efecto visual que produce.

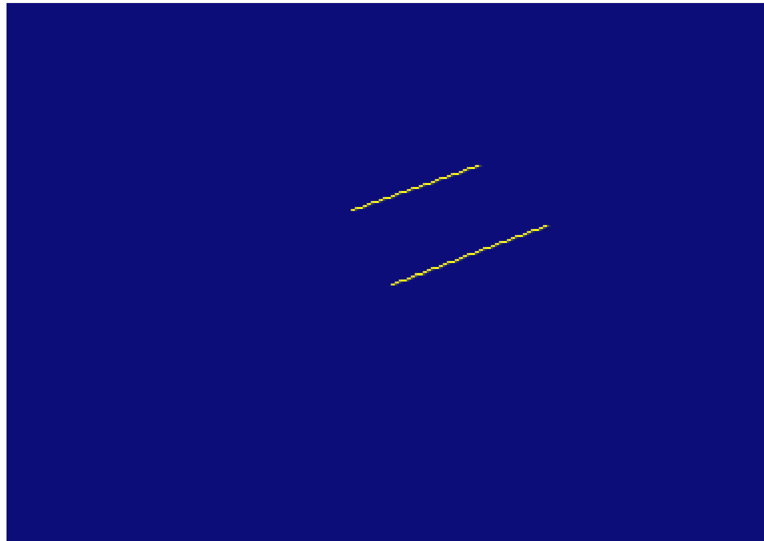
Al pulsar la barra espaciadora, la línea se dibuja permanentemente mediante la subrutina 3000. Esta subrutina pone el modo gráfico en 1 modalidad normal, dibuja la línea y vuelve a la modalidad XOR. Las coordenadas del punto se almacenan en las matrices y sus valores asignan a las variables *xant* e *yant* para servir de arranque a la recta siguiente.

Podemos modificar el programa para dar la opción de dibujar o no línea

provisional:

```
55 lineadib=0
999 REM dibujo de la recta (o no si lineadib=0)
1000 PLOT x,y,colorvisible
1005 IF lineadib=0 THEN RETURN
1010 DRAW xant,yant
1020 RETURN
2070 IF respuesta$=" " THEN GOSUB 3000:lineadib= 1
2071 IF respuesta$="1" THEN IF lineadib=0 THEN lineadib=1 ELSE lineadib=0
```

La línea 2071 permite utilizar la tecla '1' para conmutar entre ambas opciones. Cuando lineadib=0 no se dibuja la línea, terminando anticipadamente la subrutina 1000. Al ejecutar ahora el programa, podrá ver que la nueva opción le sirve también para fijar un punto sin que quede defintivamente unido por ninguna línea al anterior.



Es muy fácil hacer que se pueda borrar una línea. Vamos a limitarnos a permitir que se borre la última línea trazada, pulsando la tecla 'b':

```
2072 IF respuesta$="b" THEN GOSUB 4000
3998 REM no marcha si se trata de borrar una linea que no existe
3999 REM ya veremos como se arregla en el proximo programa
4000 x=x(con):y=y(con)
4010 con=con-1
4020 xant=x(con):yant=y(con)
4030 GOSUB 1000
4040 RETURN
```

Si le parece que la respuesta a las órdenes del teclado es algo lenta, añada las líneas siguientes:

```
1 DEFINT c,f,l,o,s,x,y
2 SPEED KEY 2,2
89 SPEED KEY 10,5
```

Ya hemos dicho antes que la utilización de enteros mejora la velocidad de los programas. La instrucción SPEED KEY va seguida de dos números que indican el tiempo que debe pasar para que comience a repetirse una tecla y el periodo de la repetición, todo ello en unidades de 1/50 segundos. Con esos dos valores se puede regular la velocidad con la que el ordendor responde a la pulsación de las teclas. Cuando se pulsa una tecla, el ordenador espera el tiempo especificado por el primer número antes de repetir el carácter. Si la tecla sigue pulsada, va repitiendo el carácter a intervalos cuya duración es la especificada por el segundo de los números. Antes de salir del programa hay que volver estos datos a sus valores normales pues, en caso contrario, sería practicamente imposible teclear correctamente los comandos posteriores.

Cuando se ha creado un dibujo que constituye una obra maestra, conviene guardarlo. Para ello hay que cambiar un poco el programa. No basta con recordar todos los puntos fijados, sino que hay que saber también qué puntos se han unido con rectas y cuáles no:

```
20 DIM x(100),y(100),l(100)
3030 con=con+1
3040 x(con)=x:y(con)=y
3045 l(con)=lineadib
3050 xant=x:yant=y
3060 RETURN
4000 x=x(con):y=y(con):lineadib=l(con)
```

Las matrices x( ) e y( ) son para las coordenadas de los puntos. La matriz l( ) sirve para saber si un punto se ha unido al anterior; para ello basta con que guarde el valor de la variable lineadib cuando se fija el punto. Si el valor guardado es O, significa que la línea no se trazó y que el punto no quedó unido al anterior. Cualquier otro valor indica que el punto quedó unido al anterior.

```
2074 IF respuesta$="i" THEN GOSUB 7000
2075 IF respuesta$="o" THEN GOSUB 8000
6000 CLG
6010 xant=1<(1):yant=y(1)
6020 FOR valor=2 TO con
6030   x=x(valor):y=y(valor)
6040   lineadib=l(valor)
6050   GOSUB 1000
6060   xant=x:yant=y
6070 NEXT
6080 x=320:y=200:lineadib=0
```

```
6090 RETURN
7000 MODE 1
7010 PRINT"Carga de datos de un fichero"
7020 INPUT"Nombre del fichero: ";fichero$
7030 OPENIN fichero$
7040 con=0
7050 WHILE NOT EOF
7060   con=con+1
7070   INPUT #9,con),y(con),l(con)
7080 WEND
7090 CLOSEIN
7100 MODE 0
7110 WINDOW 1,20,25,25
7120 xant=x(con):yant=y(con)
7130 x=xant:y=yant
7140 GOSUB 6000
7150 RETURN
8000 MODE 1
8010 PRINT"Grabacion de datos en un fichero"
8020 INPUT"Nombre del fichero: ";fichero$
8030 OPENOUT fichero$
8040 cuenta=0
8050 WHILE cuenta<=con
8060 WRITE #9,x(cuenta),y(cuenta),l(cuenta)
8070 cuenta=cuenta+1
8080 WEND
8090 CLOSEOUT
8100 MODE 0
8110 WINDOW 1,20,25,25
8120 GOSUB 6000
8130 RETURN
```

De la misma forma que se puede grabar los datos, también se los puede cargar; esto es lo que hace la subrutina 7000. No era estrictamente necesario cambiar al modo 1 para ejecutar estas rutinas, pero en modo 1 se leen mejor los mensajes.

La rutina de grabación se invoca pulsando la tecla 'o', y la de carga con `i'; hemos puesto estas para recordar input/output (o sea, entrada/salida), pero usted puede programarlo de otra forma.

Ya tenemos prácticamente completo nuestro programa de dibujo. Nos queda únicamente añadir la opción del color:

```
2070 IF respuesta$=" " THEN GOSUB 3000:lineadib=colorvisible
2071 IF respuesta$="1" THEN IF lineadib=0 THEN lineadib=colorvisible ELSE
lineadib=0
2076 IF respuesta$="c" THEN colorvisible=1+(colorvisible+1) MOD 3
3045 IF lineadib>0 THEN l(con)=colorvisible
4025 IF lineadib>0 THEN colorvisible=lineadib
```

El color crea problemas, puesto que para borrar una línea hay que dibujarla con la opción XOR y el mismo color. Antes hemos visto cómo una línea blanca, dibujada sobre una amarilla, no borraba ésta, sino que la cambiaba de color. Afortunadamente, no hay necesidad de crear una nueva matriz para almacenar los colores, pues la propia matriz I( ) sirve para ello, tanto si se desea borrar líneas como para las operaciones de grabación y carga.

Vamos a terminar el capítulo con una demostración de la flexibilidad de un programa diseñado como éste. Hemos añadido una opción que permite realizar a otra escala el dibujo que hayamos creado:

```
2073 IF respuesta$="s" THEN GOSUB 5000
5000 WINDOW 1,20,25,25
5010 INPUT"Escala: ";escala
5018 REM multiplica los valores por el factor de escala
5019 REM tomando como centro el punto donde esta el cursor
5020 FOR valor=1 TO con
5030   x(valor)=escala*(x(valor)-x)+320
5040   y(valor)=escala*(y(valor)-y)+200
5050 NEXT
5060 GOSUB 6000
5070 RETURN
```

Esta opción se invoca mediante la tecla 's', que provoca el salto a la subrutina 5000; la rutina se encarga de pedirnos la escala con la que queremos aumentar o disminuir el dibujo. Por ejemplo, una escala de 2 dibujará la figura a tamaño doble, mientras que 0.1 la reducirá a la décima parte.

La rutina aprovecha el hecho de que el Amstrad no se preocupa de saber si las

coordenadas de los comandos gráficos caen fuera de la pantalla. La rutina hace que se dibuje toda la figura, aunque sólo una parte quede dentro de la pantalla. Esta parte corresponderá a los puntos que en la versión a escala tengan coordenada X en el margen 0. . .639 y coordenada Y en el margen 0 . . .399.

Cuando se ha elegido el factor de escala, el ordenador multiplica por dicho factor las coordenadas que figuran en las matrices x(e y( ). El centro del alargamiento es el punto en el que se encontraba el cursor de dibujo en ese momento. El nuevo dibujo aparece centrado alrededor del punto (320, 200) de la pantalla, que se toma además como nueva posición del cursor.

La utilización de enteros puede resultar ventajosa para hacer más rápido un programa, pero aquí tiene la desventaja de limitar a los números enteros el factor de escala que se puede emplear.

Ejercicios

1. Añada unas pocas líneas al final del programa para devolver el modo de pantalla, la velocidad de respuesta de las teclas, etc., a sus valores habituales.
2. Introduzca instrucciones que impidan que el dibujo se salga fuera de la pantalla.
3. La rutina de borrado de líneas tiene el problema de que puede seguir intentando borrar incluso después de haber borrado todas las líneas, lo que provocará el "fracaso" del programa. Introduzca instrucciones que impidan que esto suceda.
4. Añada una rutina que permita al usuario elegir de forma cómoda los parámetros que desea para SPEED KEY.
5. Añada una rutina que vaya escribiendo en la pantalla las coordenadas del cursor en cada momento.
6. (Más difícil.) Introduzca un borrado selectivo de líneas, para que se puedan borrar otras diferentes de la última dibujada. (Puede diseñar un ciclo que vaya recorriendo todas las líneas, y preparar una tecla para seleccionar la que se desea borrar. Recuerde que además deberá reflejar este hecho en la matriz I(1, pues, en caso contrario, la recta reaparecerá cuando vuelva a cargar el dibujo desde la cinta.)

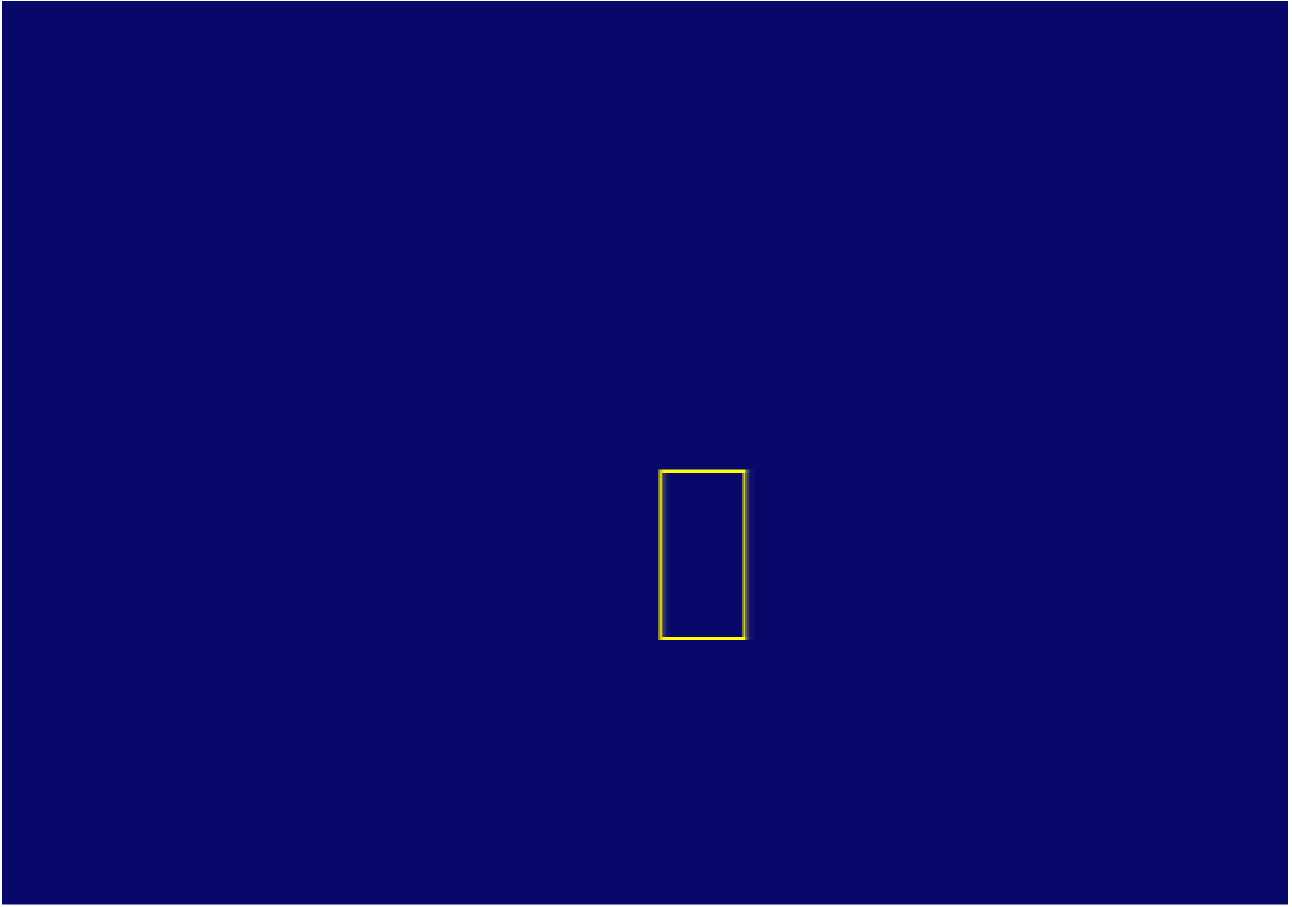
5. Animación ...

DIBUJOS LINEALES EN MOVIMIENTO

En el capítulo anterior hemos explicado cómo funciona la opción XOR y su utilidad en el dibujo y borrado de líneas. Vamos a estudiar ahora cómo podemos utilizarla para producir una animación de las figuras.

Un método para crear dibujos animados es dibujar y borrar una figura a la vez que se desplaza por la pantalla. Es un método muy primitivo, pero, para figuras simples, la velocidad del ordenador permite obtener resultados aceptables. El programa siguiente mueve un rectángulo a lo largo de la pantalla dibujándolo y borrándolo en cada posición:

```
10 MODE 1
20 x=100:y=100
30 xdistancia=50:ydistancia=100
39 REM se incrementa x en xinc cada vez que se dibuja el rectangulo
40 xinc=4
50 WHILE x<639
59   REM dibujo del rectangulo
60   color=1:GOSUB 1000
69   REM borrado del rectangulo
70   color=0:GOSUB 1000
80   x=x+xinc
90 WEND
100 END
999 REM instrucciones para dibujar el rectangulo
1000 MOVE x,y
1010 DRAWR xdistancia,0,color
1020 DRAWR 0,ydistancia
1030 DRAWR -xdistancia,0
1040 DRAWR 0,-ydistancia
1050 RETURN
```



Con pequeñas modificaciones, el rectángulo se puede mover en diagonal

```
40 xinc=4:yinc=2
80 x=x+xinc:y=y+yinc
```

Con dos líneas más se puede hacer que rebote al llegar a los bordes de la pantalla:

```
45 continue=1
50 WHILE continue=1
59   REM dibujo del rectangulo
60   color=1:GOSUB 1000
69   REM borrado del rectangulo
70   color=0:GOSUB 1000
79   REM actualizar coordenadas y ver si caen fuer a de la pantalla
80   x=x+xinc:y=y+yinc
81   IF x<0 OR x>639 THEN xinc=-xinc:x=x+2*xinc
88   IF y<0 OR y>399 THEN yinc=-yinc:y=y+2*yinc
90 WEND
```

El resultado empeora cuando interviene una figura formada por mucha líneas. Supongamos que queremos animar el dibujo de un perro hecho a base de rectas.

Para que el movimiento tenga un poco de vida, vamos a dibujar la figura en dos posiciones diferentes. El ordenador debe dibujar el perro en una posición, luego borrarlo, dibujarlo en la posición alternativa y, finalmente, borrar este dibujo. A continuación hay que actualizar las coordenadas y repetir el mismo ciclo. Para simplificar el trabajo de alternar entre los dos dibujos, lo más fácil es almacenar sus coordenadas en una matriz:

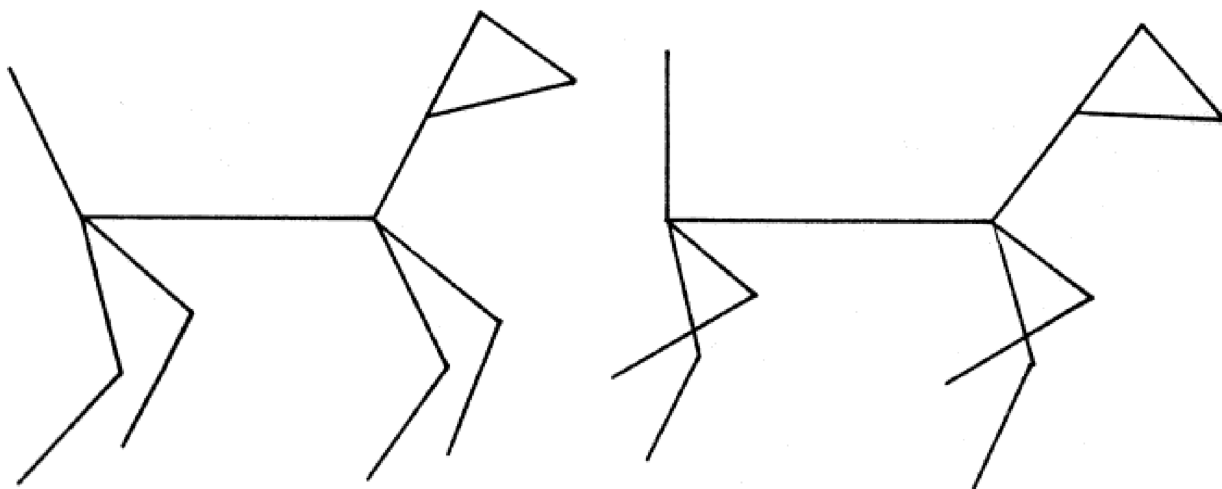
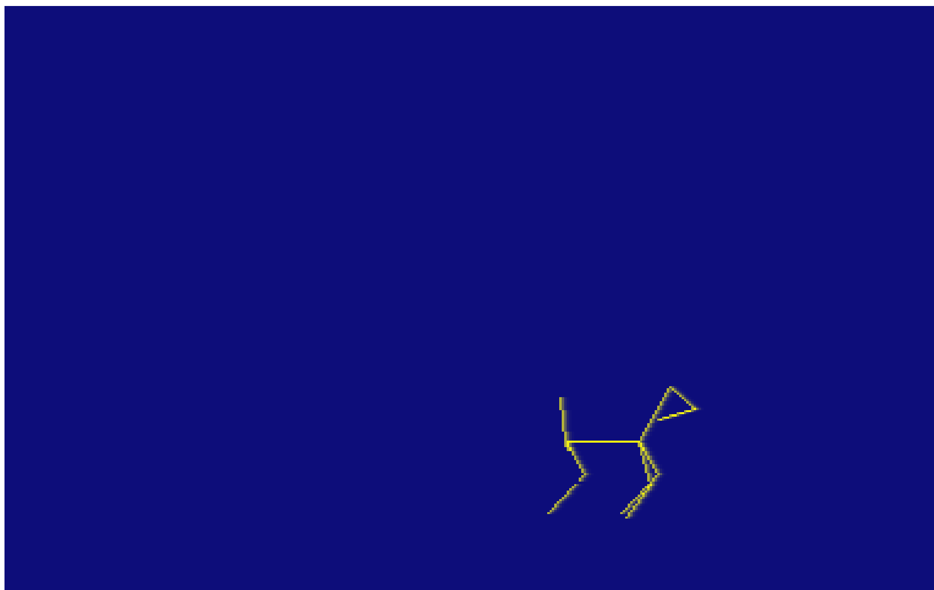


Figura 5.1. Utilización de una misma figura en dos posiciones diferentes para mejorar la ilusión de movimiento.

```
10 MODE 1
20 GOSUB 1000
30 GOSUB 2000
40 END
998 REM leer en dos conjuntos de 26 coordenadas
999 REM para dibujar dos posiciones del perro
1000 DIM x(100),y(100)
1010 FOR con=1 TO 52
1020   READ x(con),y(con)
1030   NEXT
1040 RETURN
1050 DATA 0,0,20,40,20,40,10,80,10,80,0,120,20,10,35,50,35,50,10,80,10,80,70,80
1060 DATA
60,0,80,40,80,40,70,80,80,10,95,50,95,50,70,80,70,80,90,140,90,140,110,120,110,1
20,80,110
1070 DATA 5,10,25,40,25,40,15,80,15,80,10,120,0,15,30,50,30,50,15,80,15,80,75,80
1080 DATA
75,80,85,40,85,40,65,10,75,80,90,50,90,50,60,15,75,80,100,130,100,130,120,110,12
0,110,90,100
1999 REM dibujar y borrar sucesivamente el perro
```

```
2000 xinc=0
2010 indicador=0
2020 WHILE x(1)+xinc<639
2030   IF indicador=0 THEN comienzo=1:indicador=1 ELSE comienzo=27:indicador=0
2039   REM dibujo del perro
2040   color=1:GOSUB 3000
2049   REM borrado del perro
2050   color=0:GOSUB 3000
2059   REM mover a la nueva posicion
2060   xinc=xinc+20
2070 WEND
2080 RETURN
2999 REM el perro se dibuja empalmando puntos
3000 FOR con=comienzo TO comienzo+25 STEP 2
3010   MOVE x(con)+xinc,y(con)
3020   DRAW x(con+1)+xinc,y(con+1),color
3030 NEXT
3040 RETURN
```



El proceso se ve muy bien, ya que la ejecución es sumamente lenta. lo que queda claro es que hace falta otro tipo de técnicas.

Podemos aprovechar una posibilidad que tiene el Amstrad de la que y hemos hablado. Se trata de la facilidad para cambiar la gama de color(con la instrucción INK. Si dibujamos una figura con una tinta que tenga el mismo color que el fondo de la pantalla (luego invisible), podemos hacer que aparezca repentinamente dando a la tinta del dibujo cualquier color que contraste con el del fondo:

```
30 xd=50:yd=100
39 REM poner la tinta 1 con el color del fondo
40 INK 1,1
```

```
49 REM dibujar un rectangulo usando la tinta 1
50 color=1:GOSUB 1000
60 x=300: y=300
70 xd=100:yd=50
79 REM poner la tinta 2 con el color del fondo
30 INK 2,1
89 REM dibujar un rectangulo usando la tinta 2
90 color=2:GOSUB 1000
100 continue=1
109 REM repetir los dibujos hasta que se apriete
110 WHILE respuesta$<>"e"
119   REM hacer visible la tinta 1 y la tinta 2 de 1 color del fondo
120   INK 1,21
130   INK 2,1
139   REM en espera de que se pulse una tecla
140   respuesta$=""
150   WHILE respuesta$=""
160     respuesta$=LOWER$(INKEY$).
170   WEND
179   REM hacer visible la tinta 2 y la tinta 1 de 1 color del fondo
160   INK 1,1
190   INK 2,24
199   REM en espera de que se pulse una tecla
200   respuesta$=""
210   WHILE respuesta$=""
220     respuesta$=LOWER$(INKEY$)
230   WEND
240 WEND
249 REM tintas del color habitual
250 INK 1,24
260 INK 2,20
270 END
1000 MOVE x,y
1010 DRAWR xd,0,color
1020 DRAWR 0,yd
1030 DRAWR -xd,0
1040 DRAWR 0,-yd
1050 RETURN
```

Las figuras están ya dibujadas, lo que hace que su aparición sea instantánea. Esta idea puede servir para producir un dibujo animado a base de realizar dibujos del color del fondo y darles después un color visible. ¿Podemos animar el perro de antes con éste método? La respuesta es negativa, debido a los problemas que se producen cuando se solapan dos figuras. Para entender estos problemas, lo mejor será añadir al programa anterior las líneas:

```
59 REM esta vez los dos rectangulos se solapan
60 x=120:y=100
```

y ejecutarlo. Parte de un rectángulo desaparece allí donde las figuras se superponen. Al dar a uno de los rectángulos el color del fondo, desaparece también la parte del otro rectángulo que coincide con él.

Siempre que las líneas no se solapen, los cambios de tinta pueden proporcionar movimientos suaves y rápidos. Este método es más útil en modo O, en el que se dispone de 16 tintas. Se pueden dibujar entonces hasta 15 figuras con el color del fondo, para, sucesivamente, hacerlas aparecer y desaparecer cambiando la tinta correspondiente a la figura. Por ejemplo, el siguiente programa dibuja un rectángulo que se estira y se encoge:

```
10 MODE 0
19 REM posicion de comienzo de la esquina del rectangulo
20 xcom=260:ycom=180
29 REM loggitud de los lados
30 longitudx=20:longitudy=20
39 REM diferencia del tamano de los sucesivos rectangulos
40 incx=8:incy=6
50 tintacom=1:tintafin=15
60 GOSUB 1000
70 GOSUB 2000
80 END
998 REM dibujar 15 rectangulos del color del fondo
999 REM uno dentro de otro
1000 FOR con=tintacom TO tintafin
1010   INK con,1
1020   movex=con*incx
1030   movey=con*incy
1040   ladox=longitudx+2*movex
1050   ladoy=longitudy+2*movey
1060   MOVE xcom-movex,ycom-movey
1070   DRAWR ladox,0,con
1080   DRAWR 0,ladoy
1090   DRAWR -ladox,0
1100   DRAWR 0,-ladoy
1110 NEXT
1120 RETURN
1999 REM ciclo de cambio del color de las tintas para que aparezca un reetangulo
cada vez 2000 continue=1
2010 tintacom=1:tintasig=2
2019 REM continuar indefinidamente
2020 WHILE continue=1
2029   REM se espera la pulsacion de una tecla 2030 respuesta$=""
2040   WHILE respuesta$=""
```

```
2050     respuesta$=INKEY$
2060   WEND
2069   REM cambiar el rectangulo anterior al color del fondo
2070   INK tintacom,1
2079   REM cambiar el siguiente rectangulo a color visible
2080   INK tintasig,24
2088   REM incrementar el numero de tinta para rep etir el ciclo de
instrucciones
2089   REM la actual pasa al color del fondo y la siguiente a color visible
2090   tintacom=(tintacom+1) MOD 16
2100   IF tintacom=0 THEN tintacom=1
2110   tintasig=(tintasig+1) MOD 16
2120   IF tintasig=0 THEN tintasig=1
2130 WEND
2140 RETURN
```

El método es especialmente eficaz cuando el dibujo animado es cíclico, como en este caso. El mayor problema es que no funciona correctamente si se solapan las distintas figuras.

Ya vimos como, con la modalidad gráfica XOR, es posible dibujar una línea sin afectar a las ya dibujadas. Añadiendo las líneas

```
15 PRINT CHR$(23)CHR$(1);
265 PRINT CHR$(23)CHR$(0);
```

al programa de los rectángulos parcialmente superpuestos, podremos comprobar si la opción XOR sirve en este caso. La modificación funciona en parte, puesto que los dos rectángulos se ven ahora completamente; pero la parte en litigio aparece con un inoportuno color rojo. Podemos entender este resultado si reflexionamos sobre la representación de los colores en el modo 1.

En el modo 1 existen sólo cuatro tintas diferentes. La razón es que cada tinta emplea un código binario de dos bits y que hay exactamente cuatro combinaciones posibles de dos bits. Los códigos de las tintas nunca cambian, aunque se pueden cambiar los colores de las tintas y, por lo tanto, el resultado efectivo. Por ejemplo, se puede dibujar un punto rojo con la tinta 1 usando el comando INK 1,6.

00000000	Tinta 0 (azul)	La tinta de cada
00000001	Tinta 1 (amarillo)	punto se caracteriza
00000010	Tinta 2 (cyan intenso)	mediante un código
00000011	Tinta 3 (rojo)	de 2 bits

Figura 5.2. Códigos de 2 bits para las tintas y colores del modo 1.

En nuestro ejemplo concreto se dibujan dos rectángulos; uno con la tinta 1 (código 01) y el otro con la tinta 2 (código 10). Recuerde el efecto de XOR: si los bits que se combinan son iguales, el resultado es 0; si son distintos, es 1. Lo que ocurre en los puntos donde los rectángulos se superponen se muestra en la figura 5.3:

01	Un punto de una línea amarilla
XOR 10	superpuesto con uno de una línea cyan
11	da un punto rojo

Figura 5.3.

Un punto de una línea amarilla superpuesto con uno de una línea cyan da un punto rojo. El resultado es el código 11 de la tinta 3, o sea, la tinta roja. Por lo tanto, si hacemos que la tinta 3 sea amarilla en lugar de roja, habremos conseguido que se vea la parte superpuesta:

```
11 REM tinta 3 de color amarillo
12 INK 3,24
261 REM tinta 3 a su color normal
262 INK 3,6
```

El solapamiento no resulta ya un inconveniente. Esto forma la base de un método que produce una animación bastante suave:

1. la primera figura se dibuja en una tinta que contraste con el fondo;
2. la segunda figura se dibuja en una tinta que tenga el mismo color que la del fondo;
3. se conmutan las tintas de las dos figuras, de forma que la segunda aparezca y la primera desaparezca;
4. se borra la primera figura de la posición que ocupa;
5. se dibuja una nueva figura con la tinta de la primera;
6. se conmutan las tintas, de forma que desaparezca la segunda figura y aparezca la nueva;

... y así repetidamente hasta completar la animación. El único problema estriba en el solapamiento de (as figuras, ya que entonces hace falta dibujar y borrar una figura sin afectar a la otra. Vamos a examinar este problema de más cerca.

Supongamos que estamos en el modo 1 y que utilizamos PEN 1 y PEN 2 para las dos figuras. No necesitamos preocuparnos ahora del color de cada tinta, que podremos elegir como queramos con el comando INK. Vamos a concentrarnos en los códigos de 2 bits de las tintas y en el efecto que queremos conseguir.

Para borrar la figura dibujada con PEN 1, lo que se necesita es convertir el código 01 de su tinta en el 00, que es el de la tinta del fondo. Podemos conseguirlo haciendo que sus puntos se combinen en modalidad XOR con puntos 01. Sin embargo, los que se solapen con los de la segunda figura serán puntos 11 y, al combinarse mediante XOR con 01, se convertirán en puntos de código 10. Esto es justamente lo que necesitamos.

01	Un punto de una línea amarilla
XOR 01	al que se superpone otra línea amarilla
00	da un punto del color del fondo (azul)

Figura 5.4.

Un punto de una línea amarilla al que se superpone otra línea amarilla da un punto del color azul del fondo

11	Un punto de las dos figuras
XOR 01	al que se superpone otra línea amarilla
00	da un punto de color cyan

Figura 5.5.

Un punto de las dos figuras al que se superpone otra línea amarilla da un punto de color cyan.

Pero a pesar de que todo parece resuelto, conviene reflexionar aún sobre otra cosa. Si un punto está en la intersección de dos líneas de la misma figura y se borra la figura con XOR, el primer borrado hará desaparecer el punto, pero el segundo hará que aparezca de nuevo.

Podemos obtener el resultado correcto con una técnica algo diferente, lo que nos lleva a ver otra de las modalidades gráficas del Amstrad.

El Amstrad admite también la modalidad gráfica AND. La forma de poner en marcha esta opción es el comando PRINT CHR$(23)CHR$(2).

01001101
AND 11100100
01000100

Figura 5.6. Un ejemplo del efecto de AND.

El efecto del AND es poner un bit a 1 cuando valgan 1 los correspondientes bits de los dos códigos, y ponerlo a 0 en cualquier otro caso, como se ve en el ejemplo de la figura 5.6.

Para borrar en la modalidad AND puntos escritos con la tinta 1, lo que se puede hacer es superponerles el código 10, como en la figura 5.7. 1 resultado es correcto incluso para los puntos que se encuentran simultáneamente en ambas figuras, como muestra la figura 5.8. De la misma manera, los puntos de tinta 2 se borran superponiéndoles el código 01 el la modalidad AND, como mostramos en la figura 5.9.

01	Un punto de una línea amarilla
AND 10	al que se superpone una línea cyan
00	da un punto del color del fondo (azul)

Figura 5.7.

11	Un punto de las dos figuras
AND 10	al que se superpone una línea cyan
10	da un punto de color cyan

Figura 5.8.

10	Un punto de una línea cyan
AND 01	al que se superpone una línea amarilla
00	da un punto del color del fondo (azul)

Figura 5.9.

Vamos a ver lo que ocurre ahora con el problema de dibujar sin alterar lo que

ya está dibujado. Supongamos que estamos dibujando con la tinta 1. Los resultados que conviene obtener con los distintos puntos para realizar nuestro objetivo de no destruir lo que ya está dibujado viene descrito en la figura 5.10.

Código de la tinta con la que está dibujando el punto	Código que conviene que resulte tras dibujar encima una línea de tinta 1.
00000000 00000001 00000010 00000011	00000001 00000001 00000011 00000011

Figura 5.10.

Está claro que ni la modalidad normal, ni la XOR ni la AND sirven para este objetivo. Pero existe otra opción de la que aún no hemos hablado: la modalidad OR. Esta modalidad se activa con el comando PRINT CHR$(23) CHR$(3). Da como resultado un 1 cuando es 1 alguno de los dos bits que se componen, y 0 cuando ambos son 0.

01001101
OR 11100100
11101101

Figura 5.11. Un ejemplo del efecto de OR.

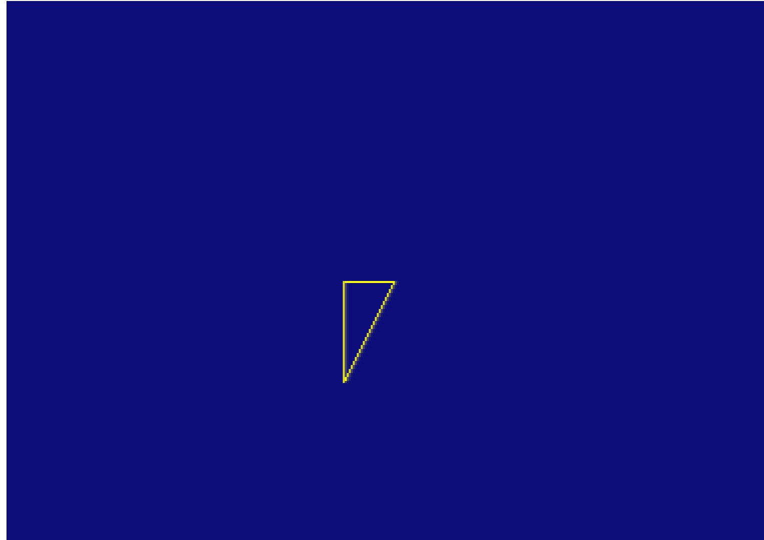
Es fácil comprobar que, al dibujar con la tinta 1 en la modalidad OR, se obtienen justamente los resultados apetecidos. Resultados similares se obtienen cuando se dibuja con la tinta 2 en la modalidad OR, como se puede ver en la figura 5.12:

00	01	10	11
OR 10	OR 10	OR 10	OR 10
10	11	10	11

Figura 5.12.

Podemos poner en práctica todos estos conocimientos para desplazar un triángulo por la pantalla:

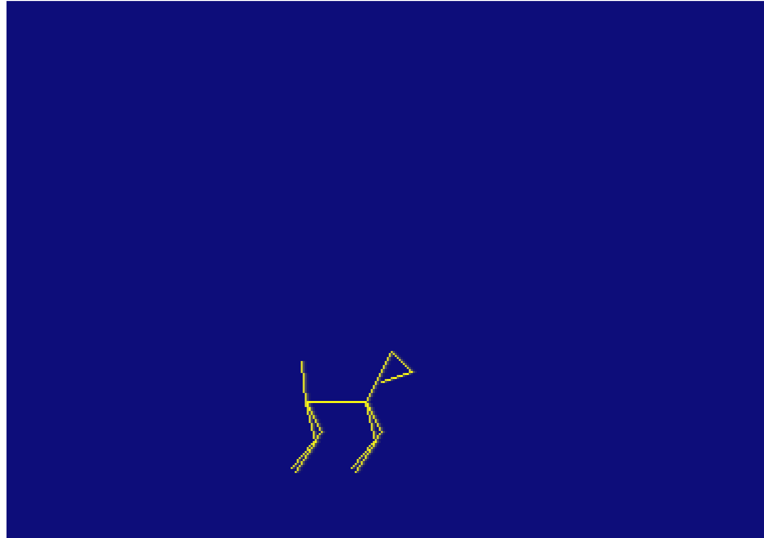
```
10 MODE 1
20 INK 3,24
30 DEFINT c,t,x,y
40 x=100:y=100
50 x1=100:y1=200
60 color=1:color1=24
70 tipo=3:tono=1
80 GOSUB 1000
90 GOSUB 2000
100 tipo=3:tono=2
110 WHILE x<639
120   x=x+4:x1=x1+4
130   GOSUB 2000
140   GOSUB 1000
150   x=x-4:x1=x1-4
160   GOSUB 2000:x=x+4:x1=x1+4
170   IF tono=2 THEN tono=1 ELSE tono=2
180 WEND
189 REM tintas de los colores usuales
190 INK 1,24
200 INK 2,20
210 INK 3,6
220 END
998 REM conmutacion de las tintas: la del color del fondo pasa a visible
999 REM y la de color visible pasa al del fondo
1000 IF color=1 THEN color=24:color1=1 ELSE color=1:color1=24
1010 INK 1,color
1020 INK 2,color1
1030 RETURN
1993 REM la rutina borra y dibuja cuando el triangulo no es visible
1999 REM (o sea, cuando esta del color del fondo)
2000 PRINT CHR$(23);CHR$(tipo);
2010 MOVE x,y
2020 DRAW x1,y1,tono
2030 DRAW x1+50,y1
2040 DRAW x+50,y1
2050 DRAW x,y
2060 IF tipo=2 THEN tipo=3 ELSE tipo=2
2070 RETURN
```



La línea 150 hace retroceder al triángulo precedente, para borrarlo cuando el color de su tinta es invisible. La línea 1000 cambia las tintas para que las figuras aparezcan y desaparezcan. La 2060 intercambia las opciones OR y AND. También se puede utilizar la misma técnica para hacer que se mueva nuestro antiguo perro:

```
10 MODE 1
14 REM para mayor velocidad se utilizan numeros enteros
15 DEFINT c,s,t,x,y
20 GOSUB 1000
30 GOSUB 2000
40 PRINT CHR$(23)CHR$(0);
50 END
998 REM lectura en dos conjuntos de 26 coordenadas
999 REM para dibujar dos posiciones diferentes d el perro
1000 DIM x(100),y(100)
1010 FOR con=1 TO 52
1020   READ x(con),y(con)
1030 NEXT
1040 RETURN
1050 DATA 0,0,20,40,20,40,10,80,10,80,0,120,20,10,35,50,35,50,10,80,10,80,70,80
1060 DATA
60,0,80,40,80,40,70,80,80,10,95,50,95,50,70,80,70,80,90,140,90,140,110,120,110,1
20,80, 110
1070 DATA 5,10,25,40,25,40,15,80,15,80,10,120,0,15,30,50,30,50,15,80,15,80,75,80
1080 DATA
75,80,85,40,85,40,65,10,75,80,90,50,90,50,60,15,75,80,100,130,100,130,120,110,12
0,110, 90,100
1999 REM la tinta 3 con el color que interesa para XOR
2000 INK 3,24
2010 color=1:color1=24
```

```
2020 tipo=3:tono=1
2029 REM dibujo de la figura en la posicion inicial
2030 GOSUB 4000
2040 GOSUB 5000
2050 tipo=3:tono=2
2060 xinc=0
2070 WHILE x(1)+xinc<639
2079   REM se actualiza xinc para la nueva figura
2080   xinc=xinc+20
2089   REM dibujo de la nueva figura en el color del fondo
2090   GOSUB 5000
2099   REM conmutacion de las tintas
2100   GOSUB 4000
2109   REM se borra la figura previa cuándo esta del color del fondo
2110   xinc=xinc-20
2120   GOSUB 5000
2129   REM se actualiza xinc para la figura actual
2130   xinc=xinc+20
2140   IF tono=2 THEN tono=1 ELSE tono=2
2150 WEND
2160 RETURN
2999 REM el perro se dibuja empalmando puntos
3000 FOR con=comienzo TO comienzo+25 STEP 2
3010   MOVE x(con)+xinc,y(con)
3020   DRAW x(con+1)+xinc,y(con+1),tono
3030 NEXT
3040 RETURN
3999 REM conmutacion de colores
4000 IF color=1 THEN color=24:color1=1 ELSE color=1:color1=24
4010 INK 1,color
4020 INK 2,color1
4030 RETURN
4999 REM cambio del modo grafico entre dibujo y borrado
5000 PRINT CHR$(23)CHR$(tipo);
5010 IF tipo+tono=4 THEN comienzo=1 ELSE comienzo=27
5020 GOSUB 3000
5030 IF tipo=2 THEN tipo=3 ELSE tipo=2
5040 RETURN
```



La animación que se obtiene así es bastante suave.

Ejercicios

1. Dibuje una serie de circunferencias iguales en diferentes posiciones y con diferentes tintas, de manera que, al cambiar los colores de las tintas entre el del fondo y el color visible, se reproduzca el bote de una pelota.
2. Construya, utilizando rectas, dos figuras de un gimnasta que realice un ejercicio con los brazos, animando entonces el movimiento mediante el empleo de tintas.

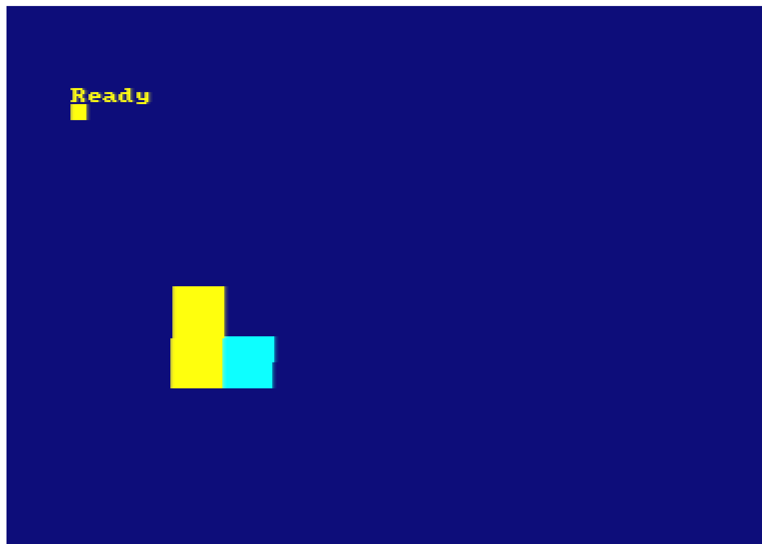
ORDEN DE PRIORIDAD DE LOS COLORES

Vamos a ver otra consecuencia de la manipulación de tintas. El programa siguiente dibuja dos rectángulos que se solapan parcialmente; esta vez se trata de rectángulos rellenos de color, uno amarillo y otro azul:

```
10 MODE 1
19 REM hacer amarillo el color cercano
20 INK 3,24
30 PRINT CHR$(23)CHR$(1);
40 x=100:y=100
50 xd=50:yd=100
60 INK 1,24
70 color=1:GOSUB 1000
80 x=100:y=100
90 xd=100:yd=50
100 INK 2,20
110 color=2:GOSUB 1000
120 PRINT CHR$(23)CHR$(0);
```

```
170 END
999 REM rectangulo relleno de color
1000 FOR xcord=x TO x+xd STEP 2
1010   MOVE xcord,y
1020   DRAWR 3,yd,color
1030 NEXT
1040 RETURN
```

Al ejecutar el programa no le sorprenderá el hecho de que uno de los rectángulos tape parte del otro. Lo que le sorprenderá seguramente es que sea el amarillo, que se ha dibujado primero, el que tape al azul, que se ha dibujado más tarde. La responsable es la línea 20, cuya consecuencia práctica es convertir en amarilla el área común a las dos figuras. Esto significa que hemos decidido que el amarillo sea el color del primer plano y el azul el color del fondo; por eso hemos dado prioridad al amarillo.



Basta con cambiar la línea 20, haciendo que el área común sea azul, y será el rectángulo azul el que tape parte del amarillo:

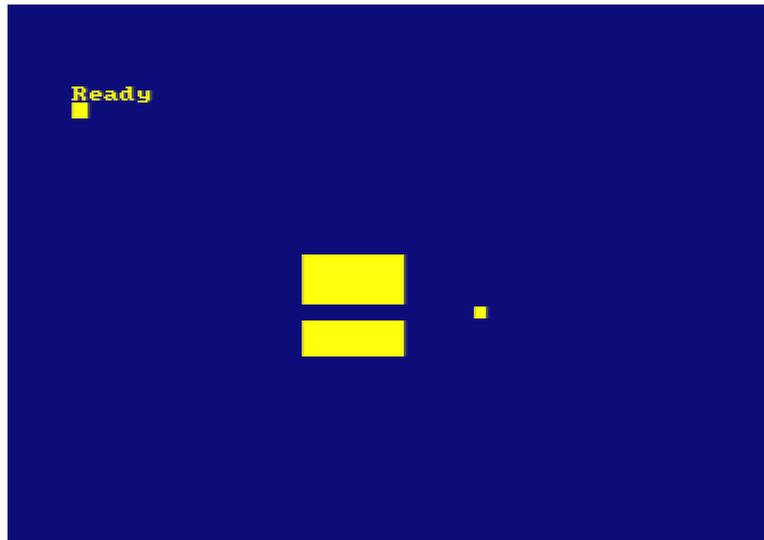
```
19 REM hacer azul el color cercano
20 INK 3,20
```



Es muy usual en los programas de juegos emplear esta posibilidad de dar prioridad a un color. Con un manejo adecuado de INK se logra que una figura pueda pasar sobre los dibujos del fondo sin borrarlos; o, lo que es todavía más impresionante, que pueda pasar por detrás de otras figuras y salir indemne por el lado contrario, como vamos a demostrar ahora.

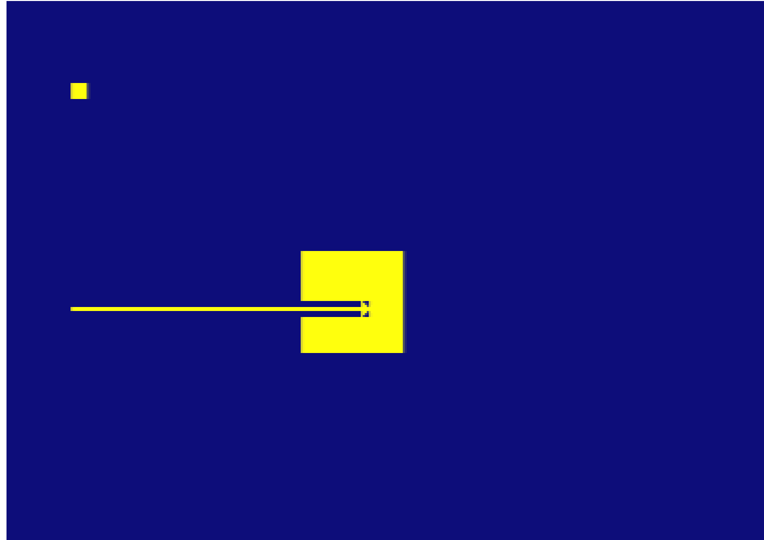
Vimos en el capítulo 2 que una de las formas más sencillas de producir un dibujo animado es utilizar la sentencia TAG e ir escribiendo un carácter en sucesivas posiciones. El efecto que esto produce cuando en la pantalla hay dibujados objetos es previsible:

```
10 MODE 1
20 x=230:y=130
29 REM dibujar y rellenar un rectangulo
30 FOR xcord=x TO x+100 STEP 2
40   MOVE xcord,y
50   DRAWR 0,100,1
60 NEXT
70 xprint=0:yprint=180
79 REM texto en el cursor grafico
80 TAG
89 REM escribir el caracter en las sucesivas posiciones
90 FOR xcord=xprint TO 400 STEP 2
100   MOVE xcord,yprint
110   PRINT CHR$(233);
120 NEXT
130 END
```



Como era lógico, el carácter elimina todo aquello sobre lo que pasa. El efecto es aun peor si el carácter es de los que no tienen un borde blanco a la izquierda:

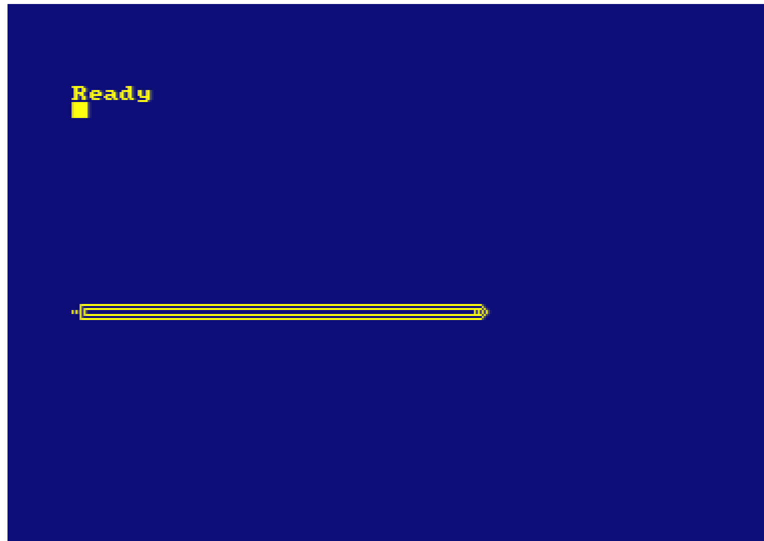
```
109 REM caracter que es una flecha sin borde blanco a la izquierda
110 PRINT CHR$(243);
```



La solución es usar TAG en combinación con la modalidad gráfica XOR. Aun así, no basta con limitarse a escribir el carácter:

```
10 MODE 1
20 x=230:y=130
29 REM dibujar y rellenar un rectangulo
30 FOR xcord=x TO x-1-100 STEP 2
40   MOVE xcord,y
50   DRAWR 0,100,1
60 NEXT
70 xprint=0:yprint=180
71 REM opcion XOR
75 PRINT CHR$(23)CHR$(1);
80 TAG
89 REM escribir el caracter en las sucesivas pos iciones
90 FOR xcord=xprint TO 400 STEP 2
100   MOVE xcord,yprint
110   PRINT CHR$(243);
120 NEXT
129 REM opcion grafica normal y modo texto normal
130 TAGOFF
140 PRINT CHR$(23)CHR$(0);
150 END
```

El resultado es aun peor. La razón del fracaso es que, al combinar en la modalidad XOR un carácter con otro igual desplazado un punto a la derecha, lo que resulta no tiene nada que ver con el carácter original.



Vamos a utilizar una flecha diseñada por nosotros con un borde blanco a la izquierda; es la de la figura 5.13.

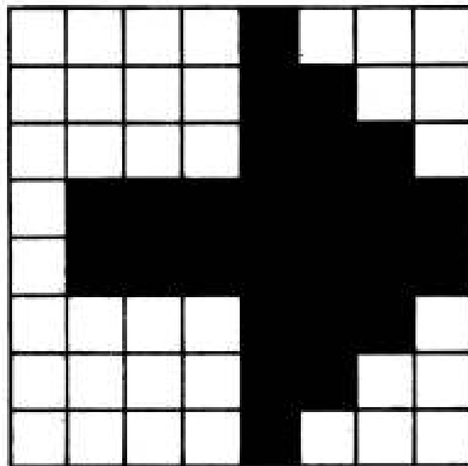


Figura 5.13. Un carácter con borde blanco a la izquierda.

Este carácter lo dibujaremos a la izquierda de la pantalla y lo iremos desplazando hacia la derecha un punto cada vez. La forma de hacerlo no va a ser, lógicamente, componerlo en la modalidad XOR con el propio carácter desplazado; eso ya sabemos que no funciona. Lo que haremos será definir un segundo carácter que, al componerlo con la flecha en la modalidad XOR, dé una copia de la flecha desplazada un punto. La figura 5.14 esquematiza lo que queremos hacer.

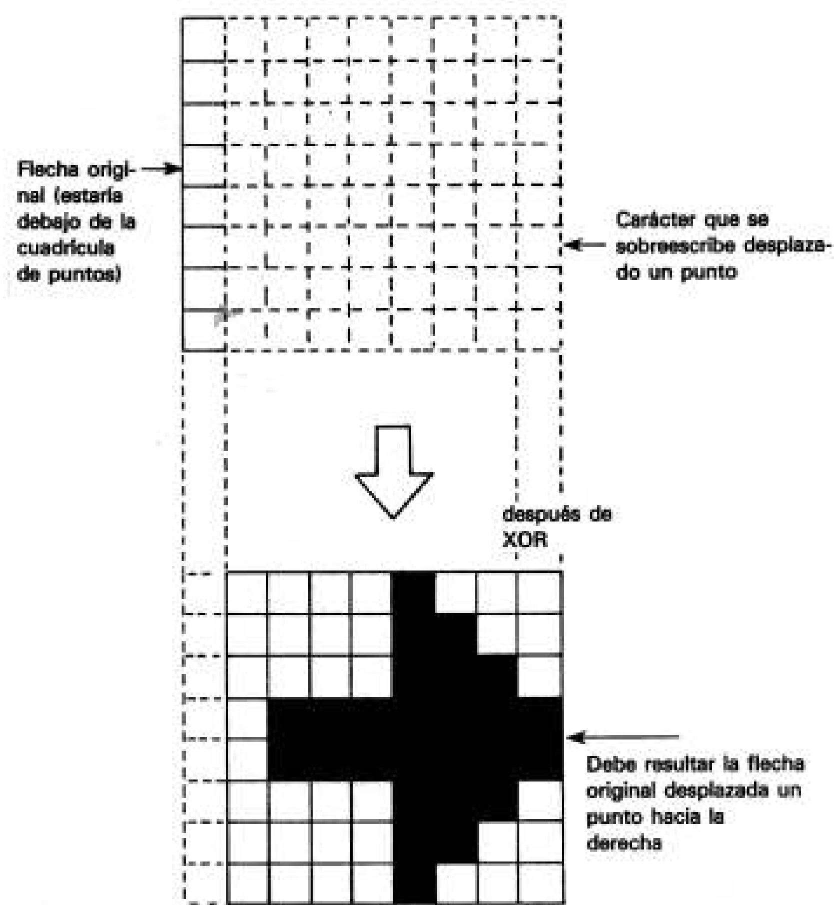


Figura 5.14. Composición con un segundo carácter en la modalidad XOR para crear una copia del primero desplazada un punto.

Es fácil buscar el segundo carácter que necesitamos. Una forma de hacerlo es proceder por tanteo fila por fila de puntos, como muestra la figura 5.15 para la primera fila.

00001000	Primera fila de la flecha (8)
XOR 00011000	compuesta con XOR con este número (24)
00001000	da la primera fila de la flecha desplazada (8)

Figura 5.15.

Pero hay un recurso más sencillo que se basa en el hecho siguiente: si se compone un número binario con un segundo mediante XOR, y el resultado se compone con el primer número, lo que resulta ahora es el segundo número. Lo que se hace entonces es componer (con XOR) la flecha con la flecha desplazada;

el resultado será el carácter que hay que componer con la flecha para obtener la desplazada. En las figuras 5.16 y 5.17 se muestra claramente este proceso.

00001000	Primera fila de la flecha (8)
XOR 00001000	compuesta con el resultado que se desea (8)
00011000	da el número con el que hay que componer (24)

Figura 5.16.

00001100	Segunda fila de la flecha (12)
XOR 00001100	compuesta con el resultado que se desea (12)
00010100	da el número con el que hay que componer (20)

00001110	Tercera fila de la flecha (14)
XOR 00001110	compuesta con el resultado que se desea (14)
00010010	da el número con el que hay que componer (18)

(y lo mismo para las restantes filas)

Figura 5.17.

Vamos a incorporar los dos caracteres en un programa, para demostrar cómo se realiza el desplazamiento dejando intacto el fondo:

```
84 REM definicion del caracter flecha
85 SYMBOL 240,8,12,14,127,127,14,12,0
86 REM definicion del caracter auxiliar para XOR
87 SYMBOL 241,24,20,10,129,129,18,20,24
90 FOR xcord=xprint TO 400 STEP 2
100   MOVE xcord,yprint
108   REM escribir la flecha normal en la primera posicion
109   REM y usar XOR para trasladar la flecha a la siguiente posicion
110   IF xcord=xprint THEN PRINT CHR$(240); ELSE PRINT CHR$(241);
120 NEXT
129 REM opcion grafica normal y modo texto normal
```

```
130 TAGOFF
140 PRINT CHR$(23)CHR$(0);
150 END
```



Con esto tenemos la puerta abierta a todo tipo de efectos en los programas de juegos. Combinando TAG, XOR y los cambios de color de las tintas, podemos diseñar todos los tipos de comportamiento que son necesarios en los juegos. Se puede hacer que el protagonista sea capaz de atravesar los obstáculos amarillos de la pantalla, pero no los azules; que el monstruo que le persigue atraviese sólo las barreras azules; o que nada detenga al supermonstruo cuando persigue al protagonista.

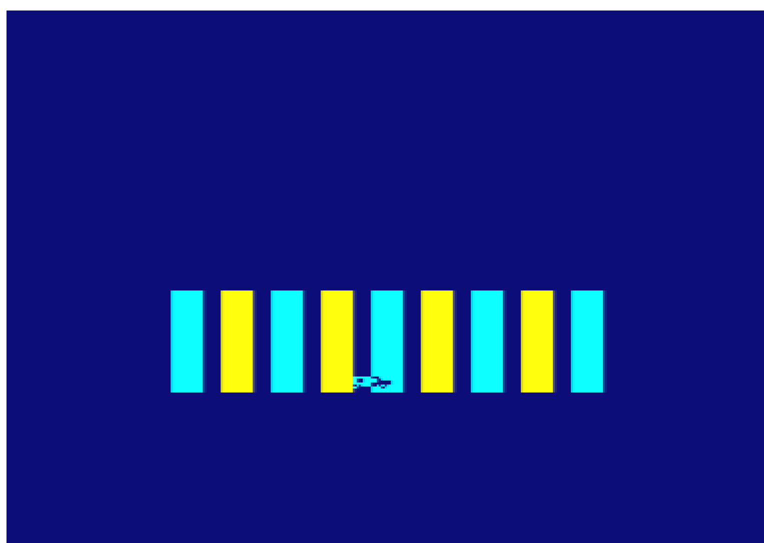
Cuando cree un carácter que se desplace, deberá recordar que debe diseñarlo con un borde blanco de un punto de ancho. De otra manera, irá dejando un rastro al desplazarse. El borde debe estar en la posición opuesta a la dirección del movimiento. La flecha de nuestro ejemplo lo llevaba a la izquierda porque debía desplazarse hacia la derecha.

Si va a desplazar un carácter en todas las direcciones, deberá diseñar un carácter auxiliar diferente por cada dirección del movimiento, empleando cada carácter al realizar el desplazamiento en la correspondiente dirección.

No hay por qué limitarse a utilizar figuras con un sólo carácter (aunque son las más sencillas). El siguiente programa crea un coche compuesto por tres caracteres, y la correspondiente figura auxiliar formada también por tres caracteres. El coche se mueve a lo largo de la pantalla, pasando detrás de los bloques amarillos y delante de los bloques de color cyan:

```
10 MODE 1
20 GOSUB 1000
30 GOSUB 2000
40 GOSUB 3000
50 TAGOFF
60 PRINT CHR$(23)CHR$(0);
```

```
70 END
998 REM las areas de solapamiento seran de la tinta 3 que hemos hecho amarilla
999 REM esto equivale a considerar mas cercano el plano del color amarillo
1000 INK 3,24
1010 y=100:color=1
1020 FOR x=100 TO 500 STEP 50
1030   GOSUB 4000
1040 NEXT
1050 RETURN
2000 SYMBOL 240,0,15,28,124,127,18,12,0
2010 SYMBOL 241,0,255,120,120,255,255,0,0
2020 SYMBOL 242,0,128,192,254,254,72,48,0
2030 SYMBOL 243,0,16,36,132,128,55,20,0
2040 SYMBOL 244,0,0,137,137,0,1,0,0
2050 SYMBOL 245,0,128,64,2,2,216,80,0
2060 coche$=CHR$(240)+CHR$(241)+CHR$(242)
2070 cochexor$=CHR$(243)+CHR$(244)+CHR$(245)
2080 RETURN
3000 PRINT CHR$(23)CHR$(1);
3010 TAG
3020 xprint=0:yprint=116
3030 FOR xcord=xprint TO 550 STEP 2
3040   MOVE xcord,yprint
3050   IF xcord=xprint THEN PRINT coche$; ELSE PRINT cochexor$;
3055   r$="":WHILE r$="":r$=INKEY$:WEND
3060 NEXT
3070 RETURN
4000 IF color=1 THEN color=2 ELSE color=1
4010 FOR xcord=x TO x+30 STEP 2
4020   MOVE xcord,y
4030   DRAWR 0,100,color
4040 NEXT
4050 RETURN
```



La línea 1000 consigue que la parte en que se solapan el coche cyan y el bloque amarillo quede de color amarillo. Si se cambia por INK 3,6 (color habitual de la tinta 3), el coche se volverá rojo al pasar por los bloques amarillos.

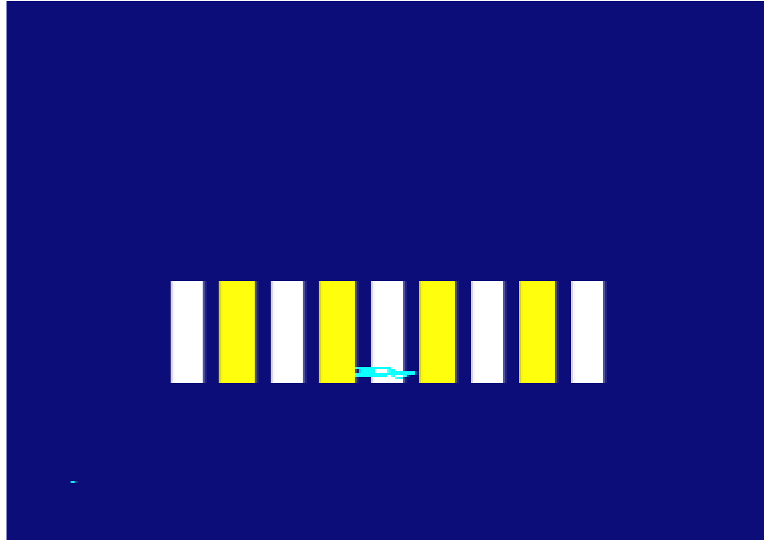
El hecho de que el coche se convierta en azul (color del fondo) cuando pasa sobre un bloque cyan deteriora bastante el buen efecto. Esto es debido a que se superponen con XOR dos figuras con la misma tinta, con lo que el resultado es O, que es la tinta del fondo.

0010	Un punto cyan
XOR 0010	solapado por otro punto cyan
0000	da un punto del color del fondo (azul)

Figura 5.18.

La solución al problema estaría en dibujar los bloques cyan con otra tinta y preparar la tinta resultante para que dé el color adecuado. El problema estriba en que el modo 1 admite 4 tintas y serían necesarias algunas más. El modo 0, con sus 16 tintas, permite hacer lo que hemos dicho:

```
10 MODE 0
998 REM tinta 3 amarilla para que resulte adecuado el resultado del solapamiento
cyan/amarillo
999 REM amarillo es el color de primer plano
1000 INK 3,24
1001 REM tinta 6 cyan para que resulte adecuado el resultado del solapamiento
cyan/blanco
1002 REM blanco es el color del fondo
1003 INK 6,20
2071 MOVE 0,0
2072 DRAW 0,0,2
3030 FOR xcord=xprint TO 550 STEP 4
4000 IF color =1 THEN color=4 ELSE color=1
4010 FOR xcord=x TO x+30 STEP 2
4020   MOVE xcord,y
4030   DRAWR 0,100,color
4040 NEXT
4050 RETURN
```



En el modo 0 cada tinta se representa por cuatro bits. El coche es de la tinta 2, y puede pasar sobre bloques amarillos (de tinta 1) y sobre bloques blancos (de tinta 4). A la vista de la figura 5.19, lo que habrá que hacer será convertir en amarilla la tinta 3 y en cyan la 6; es lo que se ha hecho en el programa en las líneas 1000 y 1003.

0010	El coche cyan
XOR 0001	sobre un bloque amarillo
0011	resulta de la tinta 3 (roja por defecto)

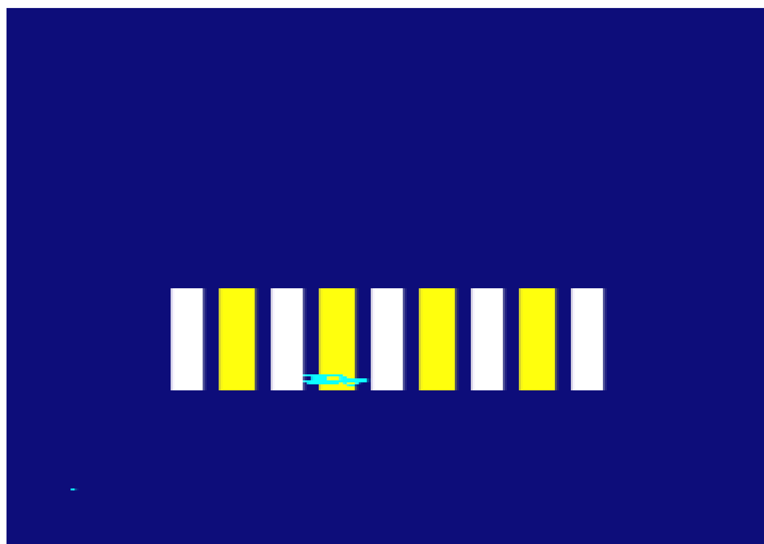
0010	El coche cyan
XOR 0100	sobre un bloque blanco
0110	resulta de la tinta 6 (azul intenso por defecto)

Figura 5.19.

También se puede hacer que el coche pase por delante de los bloques amarillos y por detrás de los blancos:

```
998 REM tinta 3 cyan para que resulte adecuado el resultado del solapamiento
cyan/amarillo
999 REM ahora amarillo es el color del fondo
1000 INK 3,20
1001 REM tinta 6 blanca para que resulte adecuad o el resultado del solapamiento
cyan/blanco
```

```
1002 REM ahora blanco es el color de primer plano
1003 INK 6,26
```



Hay que advertir que surge una pequeña complicación por el hecho de que, al usar TAG, la escritura de caracteres se hace con la tinta de la pluma gráfica; esta tinta sólo se puede cambiar con los corvados gráficos. Para que el color del coche sea cyan, la tinta de la pluma gráfica debe cambiar de 4 (blanca) a 2 (cyan). Esa es la razón de las líneas 2071 y 2072, sin las cuales el coche se dibujará en blanco.

El modo 0 ofrece las mejores oportunidades para crear diferentes planos con los colores, dado el muestrario de tintas que permite. Podemos, por ejemplo, seleccionar colores de fondo, intermedios y de primer plano, como en la figura 5.20. Con esa distribución, el bit más a la derecha indica presencia de amarillo, color lejano; el siguiente bit indica presencia de cyan, color intermedio; el siguiente indica blanco, que es el color del primer plano. Otras combinaciones de bits muestran el ocultamiento de unos colores por otros.

Con la opción XOR podemos dibujar o borrar líneas de cualquier color sin alterar las superficies solapadas o las líneas ocultas de otros colores. Como ejemplo, veamos el efecto de XOR sobre una línea blanca situada en un plano próximo, que oculta una línea cyan situada en un plano medio, que a su vez oculta una línea amarilla situada al fondo. El efecto de borrar la línea blanca se ve en la figura 5.21, y el efecto de borrar la línea cyan en la figura 5.22.

0000 Azul — fondo lejano

0001 Amarillo — fondo próximo

0010 Cyan — plano medio

0011 Cyan — plano medio tapando fondo próximo

0100 Blanco — primer plano

0101 Blanco — primer plano tapando fondo próximo

0111 Blanco — primer plano tapando plano medio, que a su vez tapa fondo próximo

Figura 5.20. Creación de diferentes planos de colores, eligiendo colores adecuados para las tintas.

0111	Blanco que oculta cyan que oculta amarillo
XOR 0100	al borrar el blanco
0011	aparece el cyan

Figura 5.21.

0111	Blanco que oculta cyan que oculta amarillo
XOR 0010	al borrar el cyan
0101	queda blanco que oculta amarillo

Figura 5.22.

Es decir, se pueden borrar colores sin alterar los que estén en otros planos. El programa siguiente dibuja una superficie blanca sobre una cyan que está a su vez sobre una amarilla, permitiendo comprobar el efecto de borrar o dibujar cualquiera de estas superficies:

```
10 MODE 0
20 GOSUB 1000
30 GOSUB 2000
40 GOSUB 3000
50 PRINT CHR$(23)CHR$(0);
60 MODE 1
70 END
999 REM tintas con las colores apropiados
1000 INK 3,20
1010 INK 5,26
1020 INK 6,26
1030 INK 7,26
1040 RETURN
1999 REM dibujar tres rectangulos uno encima de otro
2000 xamarillo=100:yamarillo=40:ladoamarillo=300
2010 xcyan=150:ycyan=80:ladocyan=220
```

```
2020 xblanco=200:yblanco=120:ladoblanco=140
2029 REM opcion XOR
2030 PRINT CHR$(23)CHR$(1);
2040 color=1:x=xamarillo:y=yamarillo:lado=ladoamarillo
2050 GOSUB 4000
2060 color=2:x=xcyan:y=ycyan:lado=ladocyan
2070 GOSUB 4000
2080 color=4:x=xblanco:y=yblanco:lado=ladoblanco
2090 GOSUB 4000
2100 RETURN
3000 WINDOW 1,20,25,25
3009 REM peticion de comando
3010 WHILE respuesta$<>"e"
3020   INPUT"Comando (a/c/b/e) ",respuesta$
3029   REM dibujo o borrado del amarillo
3030   IF respuesta$="a" THEN
color=1:x=xamarillo:y=yamarillo:lado=ladoamarillo:GOSUB 4000
3039   REM dibujo o borrado del cyan
3040   IF respuesta$="c" THEN color=2:x=xcyan:y=ycyan:lado=ladocyan:GOSUB 4000
3049   REM dibujo o borrado del blanco
3050   IF respuesta$="b" THEN color=4:x=xblanco:y=yblanco:lado=ladoblanco:GOSUB
4000
3890 WEND
3900 RETURN
4000 FOR xcord=x TO x+lado STEP 4
4010   MOVE xcord,y
4020   DRAWR 0,lado,color
4030 NEXT
4040 RETURN
```



La línea 3020 solicita una letra 'a', 'c' o 'b' para borrar o dibujar la superficie amarilla, cyan o blanca ('e' es para terminar). Borrando o dibujando cualquiera de las superficies comprobará que no se alteran las demás.

Ejercicios

1. Defina un barco con uno o más caracteres, junto con los caracteres auxiliares para desplazarlo con la opción XOR a lo largo de la pantalla.
2. Mejore el programa para que el barco navegue en una masa de agua azul y por delante de un grupo de arrecifes blancos que sobresalgan del agua.
3. Añada unas colinas de hierba que se vean detrás del barco.
4. Ponga otro barco navegando en sentido contrario. Al cruzarse, uno pasará delante del otro.
5. Cambie el orden de prioridad de los colores en el programa de las tres superficies rectangulares, de manera que el amarillo esté en primer plano y el blanco al fondo. Al comienzo, sólo deberá verse el rectángulo amarillo, y el blanco deberá aparecer únicamente después de borrar los otros dos.
6. Añada un cuarto rectángulo de color diferente y que sea más cercano que los otros tres. El blanco quedará en un plano medio delantero y el cyan en un plano medio trasero. Ajuste tintas y colores para que cada uno pueda añadirse o borrarse sin alterar los demás.

6. . . . y dibujo artístico

En el capítulo 4 desarrollamos un programa que permitía al usuario dibujar en la pantalla y grabar el dibujo en un fichero para su utilización posterior. Ahora vamos a perfeccionarlo con la inclusión de posibilidades tales como dibujo y coloreado de círculos, rectángulos y otras figuras.

SELECCIÓN A PARTIR DE UN MENÚ

El programa que teníamos se gobernaba exclusivamente pulsando determinadas teclas. Pero muchos programas de este tipo funcionan mediante selección a partir de un menú dibujado en la pantalla. Se trata de un dibujo que representa las opciones que puede escoger el usuario: selección de una figura, selección de un color, escala para el dibujo, etc. (Véase la figura 6.1.)



Figura 6.1. Menú típico de un programa de dibujo

La selección se realiza llevando el cursor a la opción deseada. El ordenador toma nota de la posición del cursor y lleva a cabo la opción indicada en dicha posición. Este sistema hace más sencilla la elección, siempre que los dibujos del menú tengan un significado suficientemente claro. Con él no hace falta recordar qué tecla hay que pulsar para cambiar el color de una línea, dibujar una circunferencia, etc.

Sin embargo, hay cosas que se controlarán mediante teclas (o con los mandos de juego, que es lo mismo). Las más evidentes son los movimientos del cursor y la fijación de un punto; nosotros dejaremos también para el teclado la alternativa entre dibujar o no dibujar la línea. Para estas funciones, es fastidioso tener que perder el tiempo recorriendo el menú con el cursor.

Nuestro nuevo programa incluye la opción de seleccionar el color de las líneas (entre ocho posibles, incluyendo el del fondo), y también la de dibujar rectas, circunferencias, triángulos o rectángulos. Otra opción es la de rellenar o no rellenar una figura cerrada.

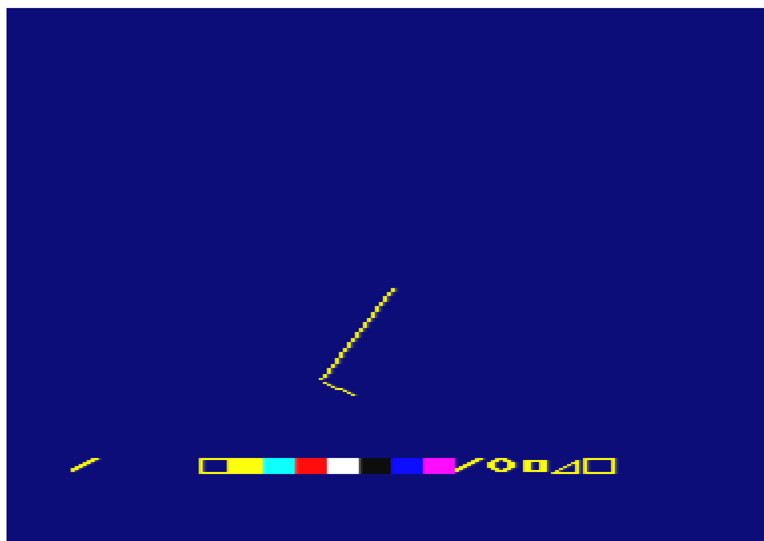
Para seleccionar la opción en el menú, hay que pulsar la barra espaciadora (que se utiliza también para fijar los puntos) cuando el cursor se encuentra sobre la opción deseada. Como resultaría difícil recordar las opciones que se han elegido, éstas se representan mediante un carácter que aparece en la esquina inferior izquierda de la pantalla, que sirve para reflejar la situación actual. El menú aparece en la parte baja de la pantalla, aunque esto es cuestión de gustos; hay quien prefiere que esté arriba o en un lateral.

Vamos a comenzar por la parte básica del programa:

```
10 MODE 0
20 GOSUB 1000
30 GOSUB 2000
40 END
1000 xant=320:yant=200
1010 x=xant:y=yant
1020 colorvisible=1
1030 lineadib=0
1037 REM imprimir el simbolo de la esquina inferior izquierda
1038 REM para que se tenga a la vista la situacion actual
1039 REM info$ es una linea amarilla al principio, que es la opcion de entrada
1040 info$=CHR$(47)
1050 LOCATE 1,24
1060 PRINT info$;
1070 menux=5:menuy=24
1079 REM definir simbolo para triangulo y para relleno/no relleno
1080 SYMBOL 240,0,2,6,10,18,34,66,254
1090 SYMBOL 241,255,129,129,129,129,129,129,255
1099 REM dibujar muestras de color de las tintas 0 a 7
1100 LOCATE menux,menuy
1110 PRINT CHR$(241);
1120 FOR color=1 TO 7
1130   PEN color
1140   PRINT CHR$(143);
1150 NEXT
1160 PEN 1
1169 REM dibujar simbolos de linea,circunferencia,rectangulo,triangulo y no
relleno
1170 PRINT CHR$(47)CHR$(79)CHR$(232)CHR$(240)CHR$(241);
1180 PRINT CHR$(23)CHR$(1);
1900 RETURN
1999 REM cursor a la posicion inicial
2000 GOSUB 3000
2009 REM repetir hasta que se pulse 'e'
2010 WHILE respuesta$<>"e"
2019   REM borrar el cursor con XOR
2020   GOSUB 3000
2029   REM leer la entrada del teclado
2030   GOSUB 4000
2039   REM dibujar el cursor
2040   GOSUB 3000
2900 WEND
2910 RETURN
2999 REM rutina de dibujo/borrado de lineas
3000 PLOT x,y,colorvisible
```

```
3010 IF lineadib=0 THEN RETURN
3020 DRAW xant,yant
3030 RETURN
3999 REM rutina para examinar el teclado y tomar la decision apropiada
4000 respuesta$=LOWER$(INKEY$)
4010 IF respuesta$="a" THEN y=y+2
4020 IF respuesta$="z" THEN y=y-2
4030 IF respuesta$="," THEN x=x-4
4040 IF respuesta$="." THEN x=x+4
4048 REM la barra de espacio fija el punto
4049 REM si la coordenada y corresponde a la zona de dibujo
4050 IF respuesta$=" " AND y>31 THEN GOSUB 5000: lineadib=colorvisible
4059 REM si no, la barra de espacio selecciona opcion del menu
4060 IF respuesta$=" " AND y<32 THEN GOSUB 6000
4069 REM conmutar entre si/no el dibujo de la li nea
4070 IF respuesta$="1" THEN IF lineadib=0 THEN lineadib=colorvisible ELSE
lineadib=0
4900 RETURN
4999 REM dibujar de manera permanente una linea
5000 PRINT CHR$(23)CHR$(0);
5010 GOSUB 3000
5020 PRINT CHR$(23)CHR$(1):
5030 xant=x:yant=y
5900 RETURN
5999 REM elegir opcion del menu; se rechaza si no esta en el menu
6000 IF x<128 OR x>543 OR y<16 THEN RETURN
6010 SOUND 7,400
6019 REM x<384 significa que la opcion es un cambio de color
6020 IF x<384 THEN colorvisible=TEST(x,y):GOSUB 7000:RETURN
6029 REM deducir opcion segun sea la coordenada
6030 IF x<416 THEN info$=CHR$(47):GOSUB 7000:RETURN
6040 IF x<448 THEN info$=CHR$(79):GOSUB 7000:RETURN
6050 IF X<480 THEN info$=CHR$(232):GOSUB 7000:RETURN
6060 IF x<512 THEN info$=CHR$(240):GOSUB 7000:RETURN
6069 REM debe ser la opcion de conmutar relleno/no relleno
6070 GOSUB 15000
6080 RETURN
7000 PEN colorvisible
7010 LOCATE 1,24
7020 PRINT info$
7030 RETURN
15000 REM a completar despues
15010 RETURN
```

Esta parte permite ya elegir las opciones, aunque, por el momento, sólo es operativa la posibilidad de dibujar rectas de diferentes colores.



El ordenador considera que se desea elegir una opción, en lugar de fijar un punto, cuando la coordenada Y del cursor es menor que 32 en el momento de pulsar la barra espaciadora. Las líneas 4050 y 4060 podrían combinarse en una sola sentencia IF THEN ELSE, pero las hemos separado para mayor claridad. La subrutina 6000 comprueba si la coordenada X corresponde al área del menú y realiza la elección dependiendo del valor de la coordenada. La 7000 dibuja el carácter info$ en la esquina inferior izquierda de la pantalla; es el carácter que sirve para recordar las opciones elegidas y cambia al seleccionar una nueva opción.

Añadir el dibujo de figuras concretas no presenta grandes problemas, pero vamos a describir cuidadosamente la forma en que se implementan estas opciones. Como ya vimos en el capítulo 3, los dibujos de circunferencias son lentos; no es aconsejable que la opción se lleve a cabo dibujando y borrando las circunferencias potenciales (con XOR), salvo que el programa quiera hacerse insoportable por su lentitud. Lo que hemos hecho es lo siguiente: cuando se ha seleccionado la opción circunferencia, el primer punto que se fije será el centro, y el siguiente, un punto de la circunferencia; por lo tanto, el radio de la circunferencia será la distancia entre ambos puntos.

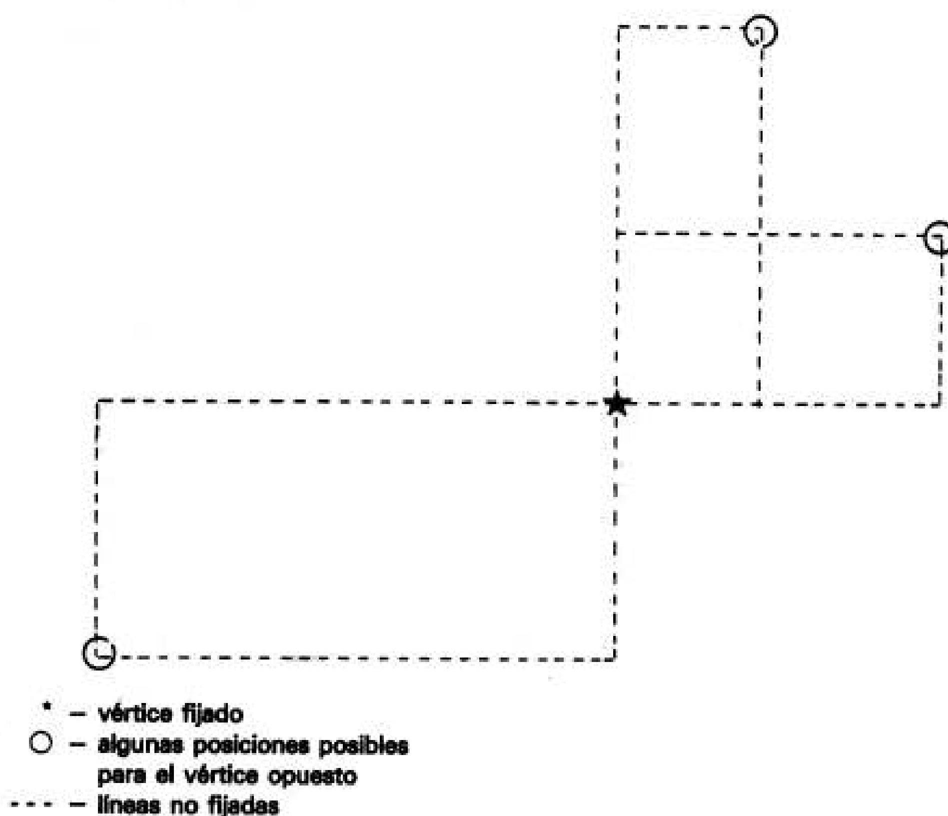


Figura 6.2. Un rectángulo queda identificado por dos vértices opuestos.

Para las opciones triángulo y rectángulo sería posible dibujar y borrar varias veces la figura, ya que intervienen pocas líneas. Las dos opciones trabajan de forma algo diferente. Cuando se han fijado dos vértices del triángulo, el tercero puede estar en cualquier parte. Sin embargo, si se fija un vértice del rectángulo de lados paralelos a los bordes de la pantalla, el resto de la figura queda perfectamente determinada en cuanto se conoce el vértice opuesto, o sea, una diagonal completa. Luego, para los rectángulos, habrá que fijar dos puntos que representarán vértices opuestos.

Cuando se está definiendo un triángulo o un rectángulo, el problema de la fijación de los puntos necesarios se resuelve como en el caso de rectas, pero el dibujo y borrado sucesivo de la figura no es lo mismo, ya que hay que trabajar con varias líneas a la vez; por eso no sirve la rutina anterior en este caso. Nosotros hemos hecho que los triángulos y rectángulos no se dibujen mientras no estén fijados todos los puntos necesarios. Esto es menos atractivo que ir dibujando el triángulo o rectángulo potencial hasta llegar al que se desea, pero esta modalidad podrá realizarla el lector como ejercicio.

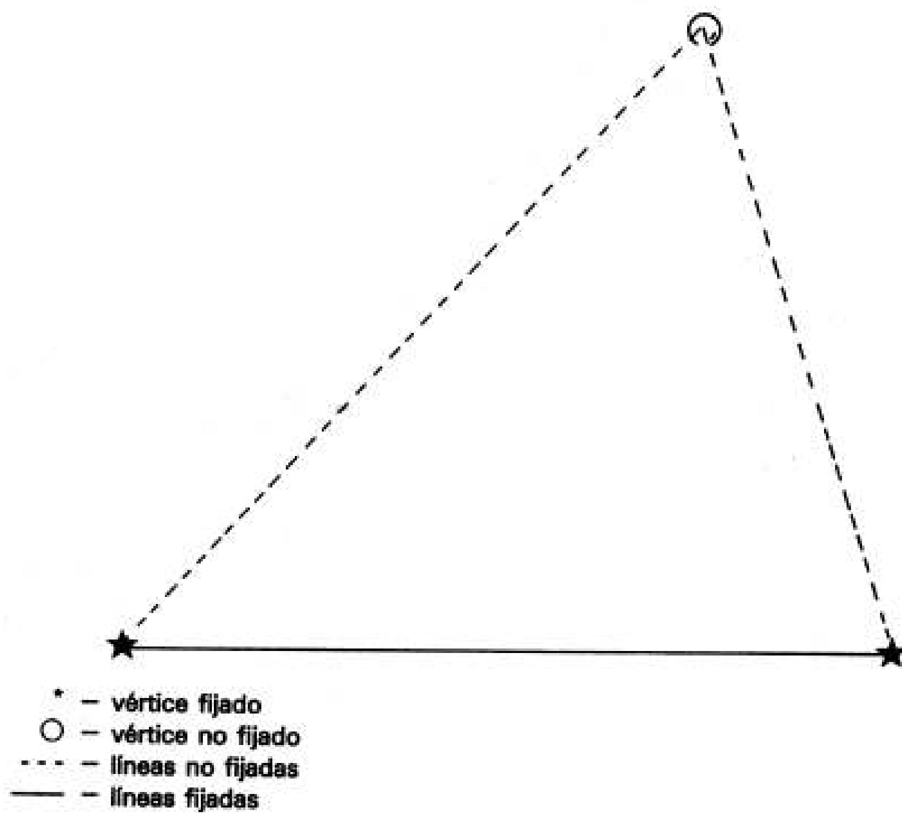
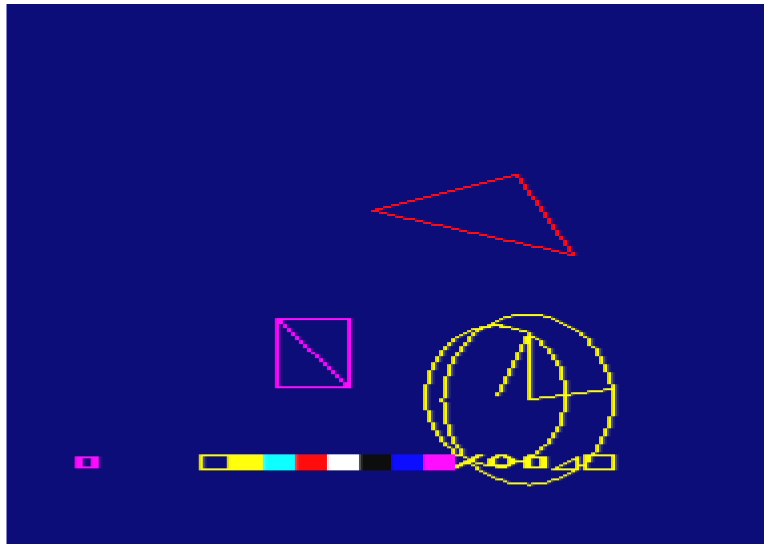


Figura 6.3. Mientras no se fije el tercer vértice de un triángulo, dos de los lados quedarán indeterminados.

```
4999 REM comprobacion de la eleccion circ/rectangulo/triangulo.
5000 PRINT CHR$(23)CHR$(0);
5001 IF circ>0 THEN GOSUB 8000
5002 IF rectangulo>0 THEN GOSUB 9000
5003 IF triangulo>0 THEN GOSUB 10000
5010 GOSUB 3000
5020 PRINT CHR$(23)CHR$(1);
5030 xant=x:yant=y
5700 RETURN
5999 REM elegir opcion del menu; se rechaza si no esta en el menu
6000 IF x<128 OR x>543 OR y<16 THEN RETURN
6010 SOUND 7,400
6019 REM x<784 significa que la opcion es un cambio de color
6020 IF x<384 THEN colorvisible=TEST(x,y):GOSUB 7000:RETURN
6029 REM deducir opcion segun sea la coordenada
6030 IF x<416 THEN info$=CHR$(47):GOSUB 7000:RETURN
6040 IF x<448 THEN info$=CHR$(79):GOSUB 7000:circ=1:RETURN
6050 IF x<490 THEN info$=CHR$(232):GOSUB 7000:rectangulo=1:RETURN
```

```
6060 IF x<512 THEN info$=CHR$(240):GOSUB 7000:triangulo=1:RETURN
6069 REM debe ser la opcion de conmutar relleno/no relleno
6070 GOSUB 15000
6080 RETURN
7000 PEN colorvisible
7010 LOCATE 1,24
7020 PRINT info$
7025 IF x<384 THEN RETURN
7029 REM opcion anulada; nueva eleccion
7030 circ=0
7040 triangulo=0
7050 rectangulo=0
7060 RETURN
7999 REM dibujo de la circunferencia: se requiere el centro y un punto de la
circunferencia
8000 IF circ=1 THEN circ=2:RETURN
8009 REM ahora tenemos los dos puntos y podemos calcular el radio
8010 xd=ABS(x-xant):yd=ABS(y-yant)
8020 radio=SQR(xd*xd+yd*yd)
8030 MOVE xant,yant+radio
8040 FOR angulo=0 TO 2*PI STEP PI/60
8050   DRAW xant+radio*SIN(angulo),yant+radio*COS(angulo)
8060 NEXT
8070 DRAW xant,yant+radio
8080 PLOT xant,yant,0
8089 REM poner el indicador de circ a 1 para la siguiente circunferencia
8090 circ=1
8100 RETURN
8999 REM rutina de dibujo del rectangulo; se requieren dos vertices
9000 IF rectangulo=1 THEN rectangulo=2:RETURN
9010 MOVE xant,yant
9020 DRAWR x-xant,0,colorvisible
9030 DRAWR 0,y-yant
9040 DRAWR xant-x,0
9050 DRAWR 0,yant-y
9059 REM poner el indicador rectangulo a 1 para el rectangulo siguiente
9060 rectangulo=1
9070 RETURN
9999 REM rutina de dibujo del triangulo; se nece sitan los tres vertices
10000 IF triangulo=1 THEN triangulo=2:x1=x:y1=y:RETURN
10010 IF triangulo=2 THEN triangulo=3:RETURN
10020 MOVE x,y
10030 DRAW xant,yant,colorvisible
10040 DRAW x1,y1
10050 DRAW x,y
10059 REM poner el indicador triangulo a 1 para el siguiente triangulo
10060 triangulo=1
```

```
10070 RETURN
15000 REM a completar despues
15010 RETURN
```



La selección de las opciones circunferencia, rectángulo o triángulo se manifiesta porque pone a 1 el correspondiente indicador (la variable cuyo nombre coincide con la opción). La rutina 7000 sirve para que los otros indicadores queden a O; en otro caso se dibujarían dos o tres figuras a la vez. La línea 7025 se incluye para que no se pongan los indicadores a 0 cuando la opción ha sido simplemente de cambio de color. Las líneas 5001 a 5003 se encargan de enviar el programa a la rutina apropiada para la realización de la correspondiente figura.

Las subrutinas 8000, 9000 y 10000, que son las encargadas de realizar estos dibujos, tienen algo en común: si el número de puntos seleccionados no completa los que necesita la figura, la rutina se limita a incrementar el indicador, pasándose a la selección de nuevos puntos. Esto es lo que hacen las líneas 8000, 9000, 10000 y 10010. El indicador sirve así al mismo tiempo para contar los puntos fijados para la figura. Dos puntos bastan para la circunferencia o el rectángulo, pero la opción triángulo necesita tres.

Cuando los puntos fijados son suficientes, se dibuja la figura y el indicador se carga con el valor 1. Esto significa que, si se ha escogido una opción como la circunferencia, por ejemplo, esta opción continuará en pie mientras no se elija otra explícitamente en el menú. Si no se tiene esto en cuenta, es fácil que se obtengan resultados inesperados al tratar de dibujar una recta mientras se encuentra activa la opción triángulo o rectángulo.

Por la forma en que está estructurado el programa, permite añadir sin dificultad otras opciones, tales como dibujo de elipses, rombos, etc.

Ejercicios

1. Aumente la gama de colores hasta 10 colores elegidos por usted.
2. Introduzca en el menú la opción de borrar toda la zona de pantalla dedicada al dibujo. Esto debe hacer que toda la pantalla tome el color del

fondo.

3. Añada una nueva opción que permita dibujar arcos. Un arco deberá quedar especificado por su centro y por los dos puntos de sus extremos.

RELLENADO DE COLOR

Pasaremos ahora a la rutina de rellenado o sombreado de las figuras cerradas. El Amstrad dispone de un comando que permite rellenar de un color grandes superficies; es el comando WINDOW. Lamentablemente, estas áreas son forzosamente rectangulares y definidas mediante coordenadas de texto, por lo que tendremos que elaborar nuestro propio método de rellenar una superficie con color. (Esto se refiere al modelo CPC464; en los siguientes modelos se dispone de la orden FILL, con la que se pueden rellenar recintos cualesquiera.)

Es evidente que, para rellenar completamente el interior de una figura, deberemos examinar cada uno de los puntos que están en su interior.

Dividiremos el problema en varias etapas, elaborando primero una rutina que rellene los puntos de un segmento de recta, y extendiéndola luego al caso de superficies.

Supongamos que hemos elegido un punto cualquiera del interior de nuestra figura. Un proceso de dos etapas nos permitirá rellenar de color todos los puntos de su misma fila. Se examina primero el punto que se encuentra a su derecha con la instrucción TEST. Si es del color del fondo, el punto se dibuja del color deseado; este punto se convierte ahora en el nuevo punto de arranque. El proceso se repite hasta que se encuentre un punto que no tenga el color del fondo, lo que significará que se ha alcanzado el borde de la figura. Este proceso está esquematizado en la figura 6.4.

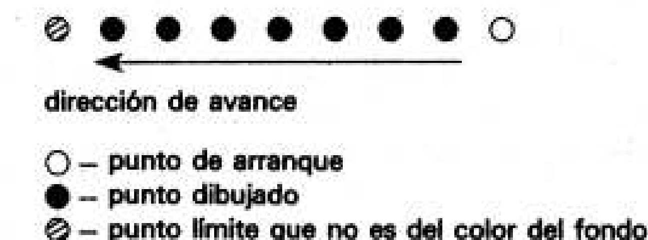


Figura 6.4.

Por este procedimiento se une el punto elegido con el borde de la figura mediante una línea horizontal. Los restantes puntos a rellenar se localizan por el mismo procedimiento, pero avanzando ahora hacia la derecha. La siguiente rutina rellena de esta manera unas cuantas líneas:

```
1 REM este es un programa independiente
2 REM pero las subrutinas
```

```
3 REM están preparadas para ser incorporadas al programa principal
10 MODE 0
19 REM dibujo de un triangulo para que sirva de ejemplo
20 MOVE 200,100
30 DRAW 450,350
40 DRAW 340,400
50 DRAW 200,100
54 REM se eligen 10 puntos al azar para rellenar rectas partiendo de ellos
55 FOR con=1 TO 10
59   REM punto al azar dentro del triangulo
60   rand=INT(RND(1)*230):xarranque=210+rand:yarranque=110+rand
70   colorvisible=1:yrell=yarranque
79   REM relleno a la izquierda del punto
80   xinc=-4:xrell=xarranque
90   GOSUB 18000
98   REM ahora a la derecha, pero no se empieza por
99   REM el propio punto sino por el que esta a su izquierda
100   xinc=4:xrell=xarranque-4
110   GOSUB 18000
120 NEXT
130 END
17999 REM comprobar punto por.punto a izquierda o derecha
18000 t=0:WHILE t=0
18010   xrell=xrell+xinc
18020   t=TEST(xrell,yrell)
18028   REM si t=0 es punto es del color del fondo
18029   REM y hay que dibujarlo
18030   IF t=0 THEN PLOT xrell,yrell,colorvisible
18040 WEND
18050 RETURN
```



La subrutina 18000 es invocada dos veces: una con incrementos de –4 para la coordenada X (avance hacia la izquierda) y otra con incrementos +4 (avance hacia la derecha). (Observe que la cuantía del incremento dependerá del modo de pantalla; en el modo 1 los incrementos serían de ±2.)

Compruebe que esta rutina funciona en cualquier caso, haciéndola trabajar con otra figura, pero cuidando de que las coordenadas xarranque e yarranque sean de un punto inferior a la figura.

¿Qué ocurre si el punto inicial está fuera de la figura? Ahora el comando TEST va a ser insuficiente por sí mismo. Yendo hacia la izquierda, podemos llegar perfectamente fuera de la pantalla; el comando TEST considera estos puntos exteriores a la pantalla como si tuvieran el color del fondo. En consecuencia no terminaríamos nunca de rellenar.

Hay varias formas de solucionar este inconveniente. Por ejemplo, podemos hacer comprobaciones sobre los valores que va tomando la variable xrell:

```
59 REM puntos dentro y fuera del triangulo
60 rand=INT(RND(1)*230):xarranque=150+rand:yarranque=110+rand
17999 REM comprobar punto por punto a izquierda o derecha
18000 t=0:WHILE t=0 AND xrell>0 AND xrell<639
```

Esto, sin embargo, requiere muchas comprobaciones y hace bastante más lento el programa. Otra posibilidad más rápida es construir un borde de color alrededor de la pantalla; al llegar al borde la rutina encontraría así un punto que no es del color del fondo, suspendiendo entonces el proceso de relleno.

Nuestra rutina para rellenar líneas ya está completa. ¿Cómo extenderla para que rellene una figura plana? Una forma es examinar los puntos que están encima y debajo de los de la línea que se va rellenando; si se encuentran puntos del color del fondo, sus coordenadas pueden almacenarse para servir posteriormente de puntos de arranque en el rellenado de nuevas líneas. Esto es lo que se representa en la figura 6.5.



Figura 6.5.

Parece en principio que sólo deberíamos ir comprobando puntos hasta encontrar uno del color del fondo. Pero ciertas formas exigen la comprobación de todos los puntos, como ocurre con la de la figura 6.6. En figuras como ésta, sólo quedarán rellenos ciertos trozos de las partes verticales si no se examinasen

todos los puntos. Comprobar todos los puntos consume mucho tiempo, por lo que deberemos elegir entre velocidad y perfección, entre rapidez con eventuales fallos o lentitud con la seguridad de un buen rellenado.

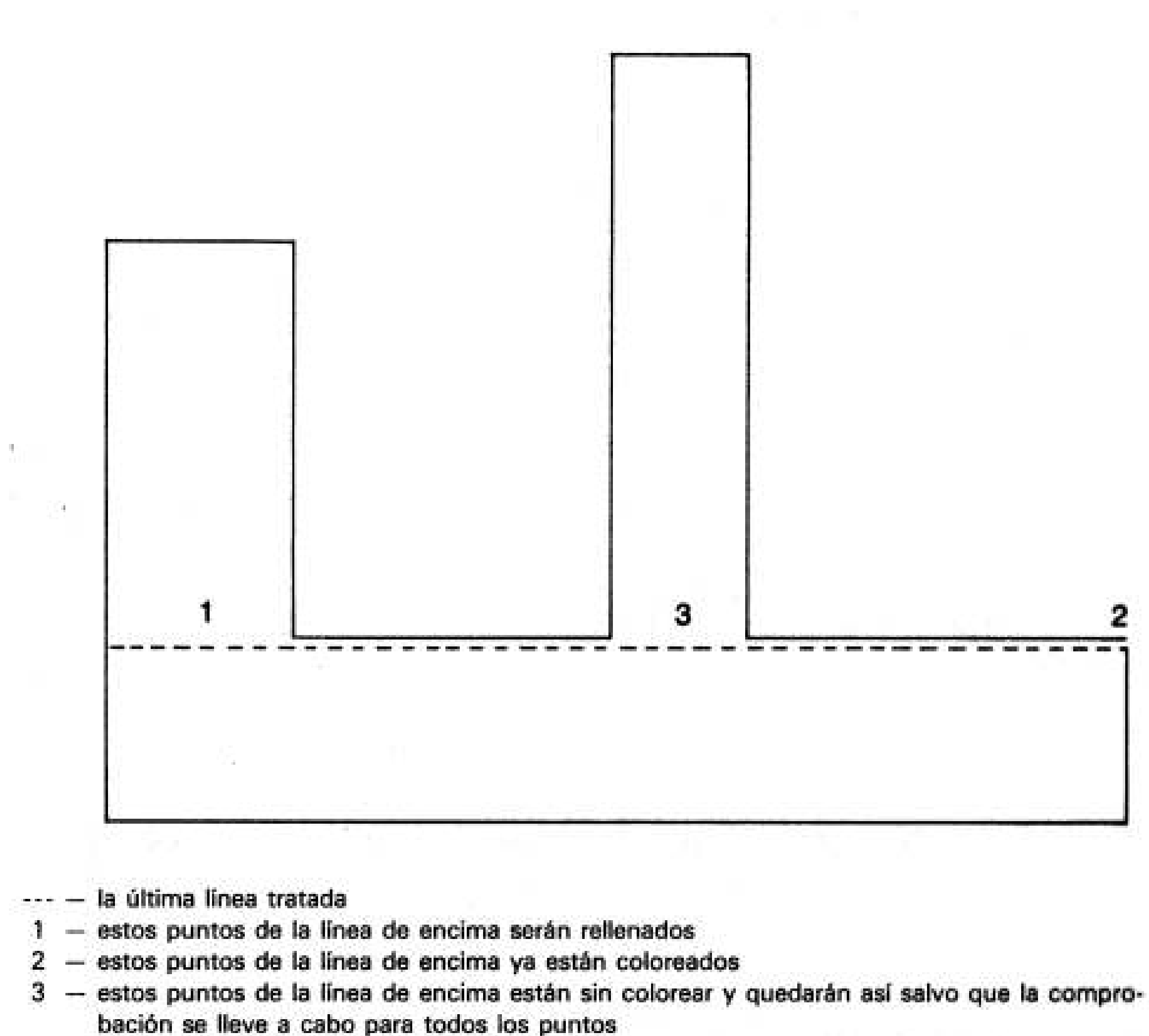


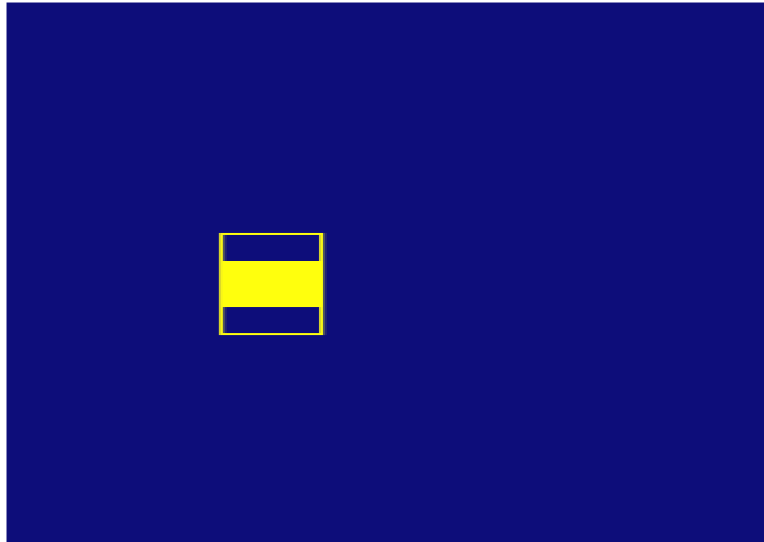
Figura 6.6.

Una rutina como la que sigue es imperfecta, pero no obstante funciona bien con buena parte de las figuras usuales examinando sólo 4 puntos:

```
10 MODE 0
14 REM matrices para las coordenadas de los punt os no coloreados
15 DIM x(300),y(300)
19 REM aqui pude icorporar su propia figura de p rueba
20 xant=150:yant=150
30 MOVE xant,yant
40 DRAWR 100,0
```

```
50 DRAWR 0,100
60 DRAWR -100,0
70 DRAWR 0,-100
78 REM x e y son las coordenadas de un punto
79 REM de dentro de la figura
80 x=200:y=200
90 colorvisible=1
100 GOSUB 15000
110 PRINT CHR$(23)CHR$(0);
999 END
15000 comienzo=2:final=1
15010 xrell=x:yrell=y
15020 x(2)=x:y(2)=y
15030 PRINT CHR$(23)CHR$(0);
15040 PLOT x,y,0
15050 GOSUB 16000
15060 GOSUB 17000
15070 PRINT CHR$(23)CHR$(1);
15080 IF circ>0 OR rectangulo>0 OR triangulo>0 THEN PLOT x,y,colorvisible
15090 RETURN
15999 REM comprobar el color del punto
16000 xactual=xrell:yactual=yrell
16010 IF TEST(xrell,yrell)<>0 THEN RETURN
16020 xinc=-4
16029 REM comprobacion del color de los puntos de la izquierda
16030 GOSUB 18000
16040 xrell=xactual-4:yrell=yactual
16049 REM comprobacion del color de los puntos de la derecha
16050 xinc=4
16060 GOSUB 18000
16070 RETURN
16999 REM se toman puntos de la lista mientras haya
17000 WHILE comienzo<>(final+1) MOD 300
17010   xrell=x(comienzo):yrell=y(comienzo)
17020   comienzo=(comienzo+1) MOD 300
17030   GOSUB 16000
17040 WEND
17050 RETURN
17999 REM comprobacion punto por punto a izquierda o derecha
18000 t=0:WHILE t=0
18010   xrell=xrell+xinc
18020   t=TEST(xrell,yrell)
18030   IF t=0 THEN PLOT xrell,yrell,colorvisible
18040 WEND
18050 xrell=xrell-xinc:yrell=yrell-2
18059 REM si el punto de la linea de abajo no esta coloreado, se almacena
18060 IF TEST(xrell,yrell)=0 THEN GOSUB 19000
```

```
18070 yrell=yrell+4
18079 REM si el punto de la linea de arriba no esta coloreado, se almacena
18080 IF TEST(xrell,yrell)=0 THEN GOSUB 19000
18090 RETURN
18999 REM almacenamiento de las coordenadas del punto no coloreado
19000 final=(final+1) MOD 300
19010 x(final)=xrell:y(final)=yrell
19020 RETURN
```



Pruebe el programa con diferentes figuras. Los puntos que son examinados son los que se encuentran en diagonal encima y debajo de los puntos finales de la línea que se está tratando. Las coordenadas de los nuevos puntos a rellenar se almacenan en las matrices x( ) e y(1. Las variables puntero comienzo y final sirven para indicar posiciones en estas matrices; comienzo indica el elemento de las matrices que contiene las coordenadas del siguiente punto del que arrancará el proceso; final indica el primer elemento de la matriz que está todavía libre.

Por ejemplo, después de rellenar la primera línea, es muy probable que los cuatro puntos que se examinan en las líneas superior e inferior sean del color del fondo; sus coordenadas se almacenarán en x( ) e y( ). La subrutina 17000 considera sucesivamente cada punto de los que están almacenados en la matriz, y la 16000 examina los puntos que están a la izquierda y a la derecha del punto elegido. Los puntos se colorean con la subrutina 18000 (línea 18030) mientras no se encuentre uno del color del fondo. Las líneas 18050-18080 comprueban el color de los puntos situados en diagonal respecto al punto límite, y la subrutina 19000 los almacena en las matrices si son del color del fondo.

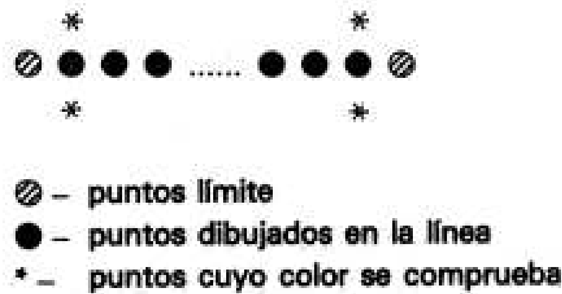


Figura 6.7.

Las líneas 17000 y 19000 llevan el operador MOD porque las matrices se tratan en forma circular. No sabemos a priori cuantos puntos deberá almacenar el ordenador, y 300 parece un límite razonable. Si la figura es grande y complicada habrá que almacenar más puntos que éstos. Lo mejor entonces es no perder los últimos puntos, sino almacenarlos en los lugares correspondientes a puntos que ya habrán sido seguramente tratados.

Esta rutina consigue rellenar un 90% de la superficie de los círculos, pero los polos superior e inferior no los rellena debido a que los arcos de dicha zona están formados por pequeños segmentos horizontales. Para nuestra rutina esto causa el mismo efecto que una línea horizontal completa. Se puede resolver este problema, con un pequeño deterioro en la velocidad, examinando dos puntos más:

```
19 REM rectangulo con un brazo vertical
20 xact=150:yact=150
30 MOVE xact,yact
40 DRAWR 100,0
50 DRAWR 0,100
55 DRAWR -10,0
56 DRAWR 0,50
57 DRAWR -60,0
58 DRAWR 0,-50
60 DRAWR -30,0
70 DRAWR 0,-100
16000 xactual=xrell:yactual=yrell
16010 IF TEST(xrell,yrell)<>0 THEN RETURN
16020 zinc=-4
16029 REM comprobacíon del color de los puntos de la izquierda
16030 GOSUB 18000
16031 REM coordenadas del extremo izquierdo de la linea
16032 xizq=xrell
16040 xrell=xactual-4:yrell=yactual
16049 REM comprobacion del color de los puntos de la derecha
16050 xinc=4 16060 GOSUB 18000
16061 REM coordenadas del extremo derecho de la linea
16062 xder=xrell
```

```
16063 REM calcular el punto medio de la linea
16064 REM y comprobar el color de los puntos de arriba y abajo
16065 xrell=(xizq+xder)/2:yrell=yactual-2
16066 IF TEST(xrell,yrell)=0 THEN GOSUB 19000
16067 yrell=yactual+2
16068 IF TEST(xrell,yrell)=0 THEN GOSUB 19000
16069 REM no es perfecto pero es rapido
16070 RETURN
```

De esta manera, los círculos se rellenan sin problemas; sólo hay fallos en figuras con trozos de forma vertical unidos al trazo principal por una recta horizontal. La posición del punto inicial influye mucho sobre cuál será la zona que será coloreada. Si se produce un fracaso en rellenar una zona desde un punto, el problema puede quedar solucionado al disponer de un punto alternativo. Por eso, nuestra rutina parece representar un buen equilibrio entre velocidad y perfección. Se puede incorporar al programa principal de la manera siguiente:

```
15 DIM x(300),y(300)
5031 IF rell=1 AND circ=0 AND rectangulo=0 AND triangulo=0 THEN GOSUB 15000
6069 REM debe ser la opcion rellenar/no rellenar
6070 IF rell=1 THEN rell=0:rell$=CHR$(241) ELSE rell=1:rell$=CHR$(233)
6080 LOCATE 17,24
6090 PRINT rell$;
6100 RETURN
8091 IF rell=1 THEN x=xant:y=yant:GOSUB 15000
9061 IF rell=1 THEN x=(x+xant)/2:y=(y+yant)/2:GOSUB 15000
10061 IF rell=1 THEN x=(x+xant+x1)/3:y=(y+yant+y1)/3:GOSUB 15000
10071 REM anadir ademas la rutina de rellenar
10072 REM desde la linea 15000 a la 19020
```

Esta rutina actúa en dos formas diferentes. Si la opción de rellenar está presente cuando se activa una de las opciones círculo, triángulo o rectángulo, la correspondiente figura se rellenará al mismo tiempo que se dibuja; el relleno comienza en un punto elegido por el ordenador en las líneas 8091, 9061 y 10061. Si la opción de rellenar está activa sin que lo esté ninguna de las tres anteriores, la rutina actúa a partir de la línea 5031 y sirve para rellenar la zona en cuyo interior se encuentre el primer punto que se fije. Por lo tanto, si se desea dibujar una recta, habrá que desactivar la opción de rellenar, pues en caso contrario el programa interpretará el punto que se fije como punto inicial de rellenado y comenzará a llenar la pantalla con el color fijado.

Ejercicios

1. Como el rellenado de figuras se hace dibujando puntos individuales, es fácil rellenar utilizando combinaciones de colores. Para ello basta con actuar sobre el color que se emplea en la línea 18030. Añada al menú la opción de

rellenar una figura con una mezcla de dos colores elegidos entre los del menú.

2. Un inconveniente del programa es que el rellenado se detiene al encontrar un punto de cualquier color que no sea el del fondo. Esto hace imposible rellenar figuras dibujadas sobre áreas ya coloreadas. Así, por ejemplo, no se podría dibujar una puerta azul sobre una casa pintada de rojo. Cambie el programa para que el rellenado se detenga solamente al encontrarse un punto de su propio color.
3. Modifique la rutina de rellenado, en la forma que ya hemos indicado, para que trabaje perfectamente con cualquier figura.

CÓMO ARCHIVAR SU OBRA MAESTRA

Ahora que sabemos realizar dibujos de colores, puede ser interesante poder archivarlos. En el capítulo 4 lo que hacíamos era almacenar las coordenadas de los puntos y otras informaciones en matrices que posteriormente se grababan en un fichero. Ahora podemos hacer lo mismo, aunque el programa requerirá modificaciones. Necesitaremos conocer, por ejemplo, si los puntos han servido para la opción circunferencia o la opción rectángulo, o si una figura va o no rellena.

Una posible alternativa consiste en grabar una copia de toda la pantalla, tal como está en un momento determinado. Si posteriormente queremos ver el dibujo o mejorarlo, basta que lo carguemos de nuevo en la pantalla y que continuemos dibujando mediante las opciones del menú. Este sistema tiene la ventaja de que puede ser implementarlo sin necesidad de alterar la parte del programa que ya tenemos.

El almacenamiento en un fichero de la pantalla es sólo un ejemplo particular de la posibilidad que tiene el Amstrad de guardar una copia de cualquier parte de la memoria. Como ya hemos dicho en otra ocasión, la pantalla no es más que una representación de una zona de memoria RAM. Haciendo una copia de esta zona, tendremos en cinta una copia de la pantalla.

El Amstrad necesita algunas informaciones para grabar una copia de una zona de la memoria: la posición en que comienza la zona y su tamaño (en bytes). Esta información también queda grabada, por lo que no se necesita recordarla cuando el fichero vuelve a cargarse en el ordenador.

Las rutinas de grabación y carga se activan pulsando las teclas 'o' (de 'output) para grabación e 'i' (de 'input') para carga:

```
4080 IF respuesta$="i" THEN GOSUB 11000
4090 IF respuesta$="o" THEN GOSUB 12000
10999 REM abrir una ventana que no estropee el dibujo
11000 WINDOW 1,20,24,25:PEN 1
11010 PRINT "Para carga"
```

```
11020 INPUT "nombre: ";dibujo$
11030 LOAD dibujo$
11040 CLS
11049 REM de nuevo la pantalla normal
11050 WINDOW 1,20,1,25
11060 GOSUB 1000
11070 RETURN
12000 WINDOW 1,20,24,25:PEN 1
12010 PRINT "Para grabacion"
12020 INPUT "nombre: ";dibujo$
12029 REM almacenamiento de la pantalla
12030 SAVE dibujo$,b,&C000,&3FCF
12040 CLS
12050 WINDOW 1,20,1,25
12055 rell=0:GOSUB 7030
12060 GOSUB 1000
12070 RETURN
```

La subrutina 12000 graba en cinta un fichero con los datos de la pantalla. Es importante que el dibujo de la pantalla no se estropee con los mensajes que deben escribirse; por eso se reserva una ventana en la parte baja de la pantalla que hace desaparecer temporalmente el menú. La línea 12030 graba el fichero: B indica un fichero de tipo binario (son los que hay que emplear para grabar zonas de memoria), &C000 es la dirección hexadecimal del comienzo de la zona de memoria que corresponde a la pantalla y &3FCF su longitud en bytes. Cuando el fichero ha sido grabado, el menú reaparece y el programa continúa. La subrutina 11000 carga un dibujo; es similar a la rutina de grabación, salvo por el hecho de que el comando LOAD de la línea 11030 es mucho más simple. Cargar un dibujo es muy espectacular: el dibujo no se carga comenzando por arriba, como se podría pensar, sino en forma de bandas horizontales que se van haciendo cada vez más gruesas hasta cubrir la pantalla. Esto refleja la complejidad de la representación de la pantalla en la memoria, donde posiciones sucesivas de la memoria pueden corresponder a zonas distantes de la pantalla.

Ejercicios

1. Añada un 'zoom' a las opciones posibles, de manera que el dibujo pueda ser cambiado a una escala diferente. (Para mejorar la velocidad puede venir bien no rellenar las figuras.)
2. Añada una rutina que permita teclear un texto y colocarlo en la posición deseada de la pantalla.
3. Modifique el programa para que admita cualquiera de los colores no parpadeantes del modo 0.

7. Transformaciones

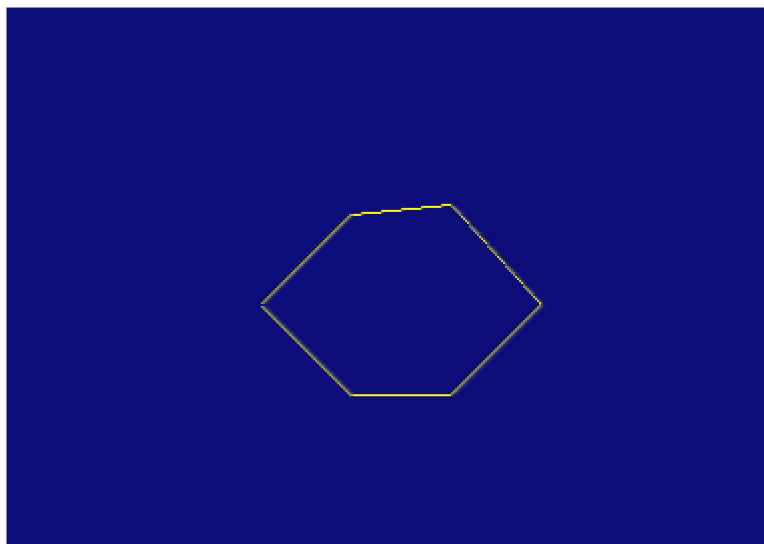
TRANSFORMACIONES DE UNA FIGURA

En otros capítulos hemos estudiado la forma de desplazar puntos en la pantalla para situarlos en la posición más adecuada. Ahora vamos a ver cómo se desplaza, no un simple punto, sino toda una figura. Además, el movimiento no tiene por qué realizarse necesariamente a base de desplazamientos sucesivos de un punto de longitud. Es frecuente tener que realizar una serie de transformaciones de una figura.

Ya sabemos hacer la más sencilla de las transformaciones, la traslación. Consiste en el movimiento en una dirección. Hasta ahora hemos trasladado puntos y caracteres, pero es fácil adaptar lo que ya hemos hecho para trasladar figuras enteras:

```
10 MODE 2
20 DIM x(100),y(100),z(100)
69 REM cargar la figura
70 GOSUB 1000
74 REM modalidad grafica
75 PRINT CHR$(23)CHR$(1);
79 REM dibujar la figura
80 GOSUB 2000
89 REM opcion
90 GOSUB 3000
800 MODE 1
900 END
999 REM lectura de los datos de la figura
1000 READ numdelin
1010 FOR con=0 TO numdelin
1020   READ x(con),y(con),l(con)
1030 NEXT
1040 RETURN
1498 REM en nuestro ejemplo es un hexagono
1499 REM sustituya la figura y sus datos por los que usted desee
1500 DATA
6,300,100,0,400,100,1,490,190,1,400,290,1,300,280,1,210,190,1,300,100,1
1999 REM rutina de dibujar/borrar
2000 MOVE x(0),y(0)
2010 FOR con=1 TO numdelin
2020   IF l(con)>0 THEN DRAW x(con),y(con) ELSE MOVE x(con),y(con)
2030 NEXT
2040 RETURN
2993 REM rutina de seleccion de la opcion; 'e' para terminar
```

```
2994 REM 't' para la traslacion
3000 respuesta$=""
3010 WHILE respuesta$<>"e"
3020   respuesta$=LOWER$(INKEY$)
3030   IF respuesta$="t" THEN GOSUB 4000
3500 WEND
3510 RETURN
3997 REM rutina de traslacion
3999 REM la traslacion se realiza con las teclas a/z/,/. y
3999 REM la figura trasladada se fija pulsando la tecla de espacio
4000 WHILE respuesta$<>" "
4005   respuesta$=LOWER$(INKEY$)
4010   xinc=0:yinc=0
4020   IF respuesta$="a" THEN yinc=2:respuesta$="":GOSUB 4500
4030   IF respuesta$="z" THEN yinc=-2:respuesta$="":GOSUB 4500
4040   IF respuesta$="," THEN xinc=-4:respuesta$="":GOSUB 4500
4050   IF respuesta$="." THEN xinc=4:respuesta$ ="":GOSUB 4500
4100 WEND
4110 respuesta$=""
4120 RETURN
4499 REM efectua la traslacion
4500 GOSUB 2000
4510 FOR con=0 TO numdelin
4520   x=x(con):y=y(con):GOSUB 10000:x(con)=x:y(con)=y
4530 NEXT
4540 GOSUB 2000
4550 RETURN
9999 REM cambio de datos por traslacion
10000 x=x+xinc:y=y+yinc
10010 RETURN
```



La traslación es la transformación más sencilla de programar, pero la técnica

que se utiliza no sirve para otras transformaciones, como puede ser la rotación de una figura. Para una serie de transformaciones, el método más generalizado es el empleo de matrices.

Transformaciones matriciales

Podemos girar, alargar o reflejar cualquier figura multiplicando las coordenadas de todos sus puntos por una matriz apropiada. Vamos a dar un breve resumen de la multiplicación de matrices, aunque no es esencial su comprensión para la simple utilización de los programas que siguen.

Para multiplicar dos matrices se multiplican los números de cada fila de la primera matriz por los de cada columna de la segunda, tal como se explica en la figura 7.1, y se van sumando los productos obtenidos. Estas sumas de productos son los números que forman parte de la matriz que constituye el resultado. Las matrices pueden ser de tamaños diferentes a los del ejemplo, pero, para calcular la transformación de un punto de una figura, son estos tamaños de matrices los que interesan. Esto se debe a que los puntos vienen dados por dos coordenadas.

$$(3 \quad 5)\begin{pmatrix}2 & 4\\1 & 2\end{pmatrix}(3\,x\,2+5\,x\,1 \quad 3\,x\,4+5\,x\,2)=(11 \quad 22)$$

Figura 7.1.

Una serie de matrices de 2 x 2 elementos sirve, por ejemplo, para girar un punto 10 grados en el sentido de las agujas del reloj, o para alargar una figura en un factor de 1.5, etc. El resultado de multiplicar una matriz de coordenadas por una matriz de transformación es, como hemos visto en el ejemplo, otra matriz de coordenadas. Si la matriz que resulta en el ejemplo la multiplicamos por la matriz de una nueva transformación, obtenemos el resultado de aplicar a un punto las dos transformaciones sucesivas; esto es lo que hemos hecho en la figura 7.2.

$$(11 \quad 22)\begin{pmatrix}1 & 2\\3 & 2\end{pmatrix}(11\,x\,1+22\,x\,3 \quad 11\,x\,2+22\,x\,2)=(77 \quad 66)$$

Figura 7.2.

De hecho, hay una manera de agilizar el proceso. Se pueden multiplicar primero las dos matrices de transformación y utilizar el resultado para transformar el punto inicial. El resultado es el mismo que el de las dos transformaciones sucesivas, como muestra la figura 7.3.

$$\begin{pmatrix} 2 & 4 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 3 & 2 \end{pmatrix}=\begin{pmatrix} 2x1+4x3 & 2x2+4x2 \\ 1x1+2x3 & 1x2+2x2 \end{pmatrix}=\begin{pmatrix} 14 & 12 \\ 7 & 6 \end{pmatrix}$$

$$(3 \quad 5)\begin{pmatrix} 14 & 12 \\ 7 & 6 \end{pmatrix}=(3x14+5x7 \quad 3x12+5x6)=(77 \quad 66)$$

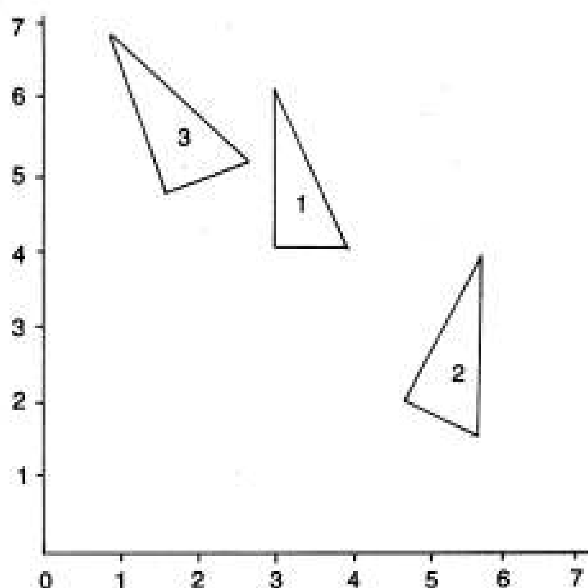
Figura 7.3.

El principal inconveniente que tiene el método matricial es que no se puede representar la traslación utilizando una matriz 2 x 2, lo que hace que este tratamiento no pueda ser totalmente general. Ya hemos visto que es posible reducir transformaciones compuestas a una sóla matriz, pero las traslaciones quedan fuera del sistema. Utilizando matrices 3 x 3 es posible representar las transformaciones, incluyendo ahora las traslaciones.

Como ya estamos acostumbrados a manejar la traslación de otra manera, vamos a quedarnos con las matrices 2 x 2, aunque tengamos que tratar la traslación de forma diferente.

Rotación

Para la rotación de un determinado ángulo hay dos matrices posibles, según que la rotación se haga en un sentido o en el otro. La figura 7.4 ilustra acerca de estas dos matrices.



El triángulo 1 se convierte en el 2 al girarlo 30 grados en el sentido de las agujas del reloj:

$$\begin{pmatrix} 3 & 4 \\ 4 & 4 \\ 3 & 6 \end{pmatrix} \begin{pmatrix} \cos 30 & -sen\,30 \\ sen\,30 & \cos 30 \end{pmatrix} = \begin{pmatrix} 4.6 & 2.0 \\ 5.5 & 1.5 \\ 5.6 & 3.7 \end{pmatrix}$$

El triángulo 1 se convierte en el 3 al girarlo 20 grados en sentido contrario a las agujas del reloj:

$$\begin{pmatrix} 3 & 4 \\ 4 & 4 \\ 3 & 6 \end{pmatrix} \begin{pmatrix} \cos 20 & sen\,20 \\ -sen\,20 & \cos 20 \end{pmatrix} = \begin{pmatrix} 1.5 & 4.8 \\ 2.4 & 5.1 \\ 0.8 & 6.7 \end{pmatrix}$$

En general, la rotación de 0 grados en el sentido del reloj viene dada por:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\alpha & -sen\,\alpha \\ sen\,\alpha & \cos\alpha \end{pmatrix} = \begin{pmatrix} x\cos\alpha + y\,sen\,\alpha & -x\,sen\,\alpha + y\cos\alpha \end{pmatrix}$$

y la rotación de 0 grados en sentido contrario al reloj, por:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\alpha & sen\,\alpha \\ -sen\,\alpha & \cos\alpha \end{pmatrix} = \begin{pmatrix} x\cos\alpha - y\,sen\,\alpha & x\,sen\,\alpha + y\cos\alpha \end{pmatrix}$$

Figura 7.4.

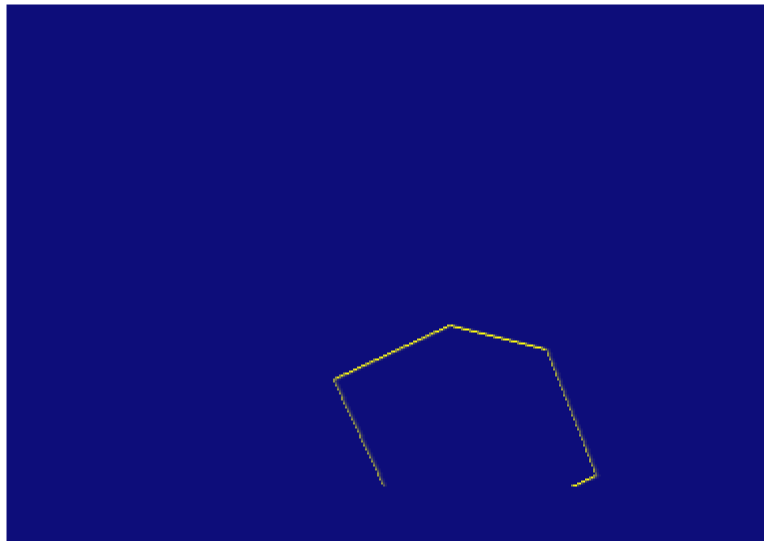
Si comparamos los resultados de la multiplicación, observamos que se diferencian solamente en los signos. Si definimos la variable rotación para que valga 1 cuando la rotación es en el sentido de las agujas del reloj, y –1 cuando el sentido es el opuesto, obtenemos una misma ecuación para el giro, independientemente de su sentido; podemos ver esto en la figura 7.5.

$$xNuevo = xViejo \cdot \cos\alpha - rotacion \cdot yViejo \cdot sen\,\alpha$$
$$yNuevo = -rotacion \cdot xViejo \cdot sen\,\alpha + yViejo \cdot \cos\alpha$$

Figura 7.5.

Se puede generalizar el programa anterior con la posibilidad de realizar también rotaciones:

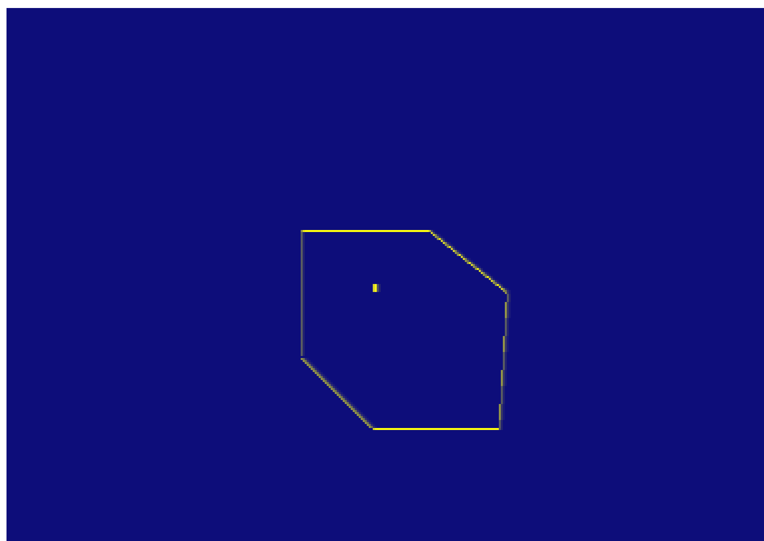
```
2995 REM 'r' para la rotacion
3040 IF respuesta$="r" THEN GOSUB 5000
4995 REM rutina de rotacion
4998 REM la rotacion es de 5 grados en el sentido del reloj
4999 REM pero puede cambiar estos datos
5000 angulo=5
5010 rotacion=1
5199 REM barra la figura anterior
5200 GOSUB 2000
5299 REM coeficientes de la transformacion
5300 DEG
5310 xtrans1=COS(angulo):ytrans2=xtrans1:xtrans2=rotacion*SIN(angulo):ytrans1=-
xtrans2
5499 REM calcula la figura girada
5500 GOSUB 11000
5599 REM dibuja la figura girada
5600 GOSUB 2000
5699 respuesta$=""
5700 RETURN
10999 REM cambio de datos por transformacion mat ricial
11000 FOR con=0 TO numdelin
11010   almacen=x(con)
11020   x(con)=x(con)*xtrans1+y(con)*xtrans2
11030   y(con)=almacen*ytrans1+y(con)*ytrans2
11040 NEXT
11100 RETURN
```



Obsérvese que la rotación se produce alrededor del punto (0, 0). Es mejor poder seleccionar el centro de giro, lo que resulta sencillo de programar:

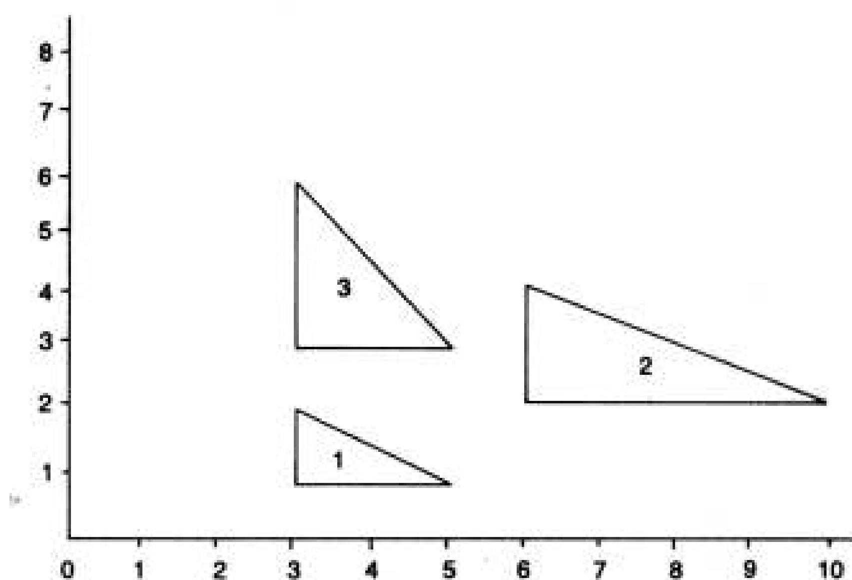
```
30 xcentro=320:ycentro=200
```

```
4996 REM el centro de la rotacion se selecciona con a/z/,/. y se fija
4997 REM pulsando la tecla de espacio
5099 REM fijacion del centro de la rotacion
5100 GOSUB 15000
5500 GOSUB 12000
11999 REM rutina para transformar las coordenadas de la figura por cualquier
transformacion matricial con centro arbitrario
12000 xinc=-xcentro;yinc=-ycentro
12010 FOR con=0 TO numdelin
12020   x=x(con):y=y(con):GOSUB 10000:x(con)=x:y(con)=y
12030 NEXT
12040 GOSUB 11000
12050 xinc=xcentro:yinc=ycentro
12060 FOR con=0 TO numdelin
12070   x=x(con):y=y(con):GOSUB 10000:x(con)=x:y(con)=y
12090 NEXT
12100 RETURN
14999 REM rutina de fijacion de un centro de la transformacion
15000 TAG
15004 REM correcciones para centrar un caracter en un punto; la izquierda debe
ser 8 en modo 0 y 4 en modo 1
15005 correccionizquierda=2:correccionarriba=4
15010 MOVE xcentro-correccionizquierda,ycentro+correccionarriba:PRINT CHR$(129);
15100 WHILE respuesta$<>" "
15110   respuesta$=LOWER$(INKEY$)
15115   xinc=0:yinc=0
15120   IF respuesta$="a" THEN yinc=2:respuesta$="":GOSUB 15500
15130   IF respuesta$="z" THEN yinc=-2:respuesta$="":GOSUB 15500
15140   IF respuesta$="," THEN xinc=-4:respuesta$="":GOSUB 15500
15150   IF respuesta$="." THEN xinc=4:respuesta$="":GOSUB 15500
15200 WEND
15210 respuesta$=""
15220 MOVE xcentro-correccionizquierda,ycentro+correccionarriba:PRINT CHR$(129);
15230 RETURN
15499 REM borrar antiguo centro y dibujar el nuevo
15500 REM
15510 MOVE xcentro-correccionizquierda,ycentro+correccionarriba:PRINT
CHR$(129);:xcentro=xcentro+xinc:ycentro=ycentro+yinc:MOVE xcentro-
correccionizquierda,ycentro+correccionarriba:PRINT CHR$(129);
15600 RETURN
```



Ampliación y reducción

Ya vimos antes que era fácil ampliar o reducir una figura, pero entonces no teníamos control sobre el centro de la ampliación. Con el método matricial podemos introducir ampliaciones diferentes en la dirección de cada eje, lo que permite estirar las figuras en la dirección de uno de los ejes; los aspectos formales se pueden ver en la figura 7.6.



El triángulo 1 se transforma en el 2 mediante:

$$\begin{pmatrix} 3 & 5 \\ 5 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 6 & 2 \\ 10 & 2 \\ 6 & 4 \end{pmatrix}$$

Con diferentes valores en la diagonal se obtiene un alargamiento en la dirección de un eje:

$$\begin{pmatrix} 3 & 5 \\ 5 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ 5 & 3 \\ 3 & 6 \end{pmatrix}$$

De manera general, los alargamientos y reducciones vienen dados por:

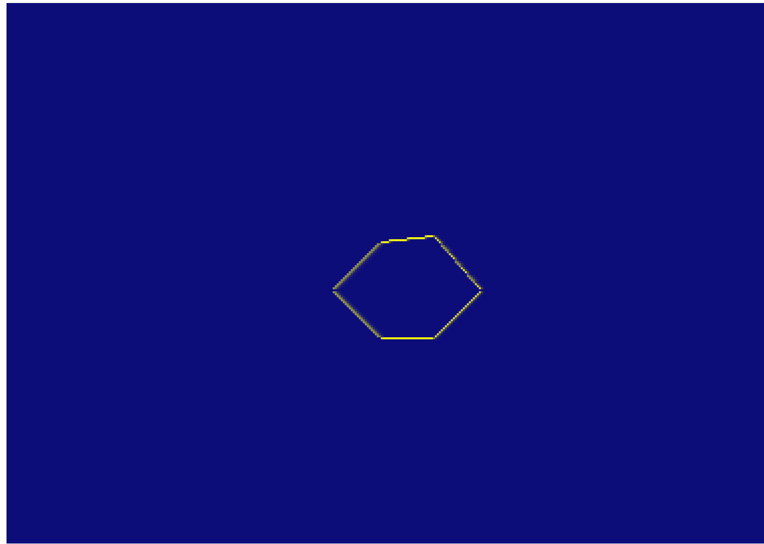
$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} n1 & 0 \\ 0 & n2 \end{pmatrix} = \begin{pmatrix} n1 \cdot x & n2 \cdot y \end{pmatrix}$$

Figura 7.6.

Es fácil añadir esta posibilidad a nuestro programa:

```
2996 REM 'g' para alargamiento
2997 REM 'p' para reduccion
3050 IF respuesta$="g" THEN GOSUB 6000
3060 IF respuesta$="p" THEN GOSUB 6100
5995 REM rutina de alargamiento o reduccion a escala
5996 REM el centro se selecciona con a/z/,/. y se fija
6010 GOTO 6200
6098 REM esta es la rutina de reduccion
6099 REM reduccion a escala 0.9 en ambos ejes; cambie el coeficiente si lo desea
6100 escalax=0.9:escalay=0.9
6199 REM fijacion del centro de alargamiento/reduccion
6200 GOSUB 15000
6299 REM borra la figura anterior
6310 GOSUB 2000
6399 REM coeficientes de la transformacion
6400 xtrans1=escalax:ytrans2=escalay:xtrans2=0:ytrans1=0
6499 REM calcula la figura transformada
6500 GOSUB 12000
6599 REM dibuja la figura transformada
6600 GOSUB 2000
```

```
6699 respuesta$=""
6700 RETURN
```



Simetría

La simetría (o reflexión) sobre el eje X o el eje Y se puede llevar a cabo con las matrices de la figura 7.7.

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} x & -y \end{pmatrix} \quad \text{(simetría de eje X)}$$

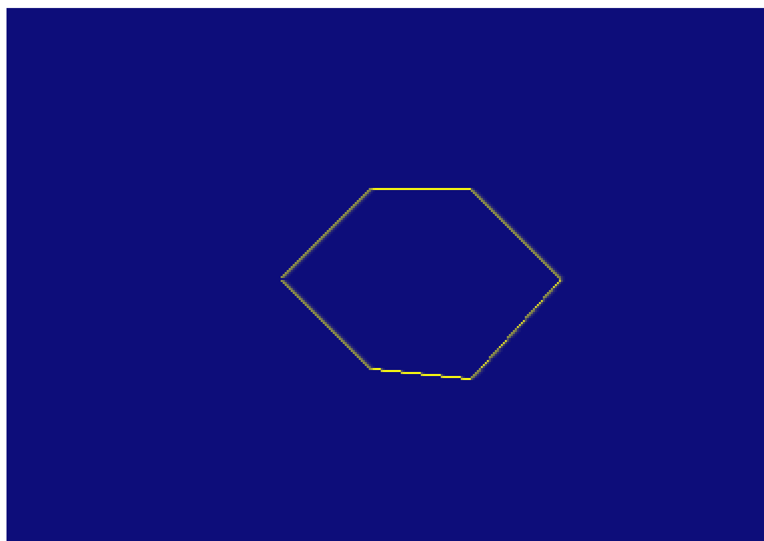
$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -x & y \end{pmatrix} \quad \text{(simetría de eje Y)}$$

Figura 7.7.

Como ocurría con la rotación, las dos matrices son en esencia una sola. Para ver el efecto de la simetría es necesario trasladar los ejes, pues de lo contrario la figura resultante se sale de la pantalla.

```
2998 REM 'f' para reflexion
3070 IF respuesta$="f" THEN GOSUB 7000
6995 REM rutina de reflexion sobre un eje
6996 REM la reflexion es sobre un eje vertical o sobre un eje horizontal
6997 REM que pasan por el punto que se desee
6998 REM el punto se selecciona con a/z/,/. y se fija
6999 REM al pulsar 'x' o 'y' , lo que selecciona ademas el eje
7000 GOSUB 16000
```

```
7099 REM coeficientes de la transformacion
7100 IF respuesta$="x" THEN xtrans1=1:ytrans2=-1 :xtrans2=0:ytrans1=0 ELSE
xtrans1=-1:ytrans2=1:xtrans2=0:ytrans1=0
7199 REM borra la figura anterior
7200 GOSUB 2000
7299 REM calcula la figura transformada
7300 GOSUB 12000
7399 REM dibujo de la figura reflejada
7400 GOSUB 2000
7499 respuesta$=""
7500 RETURN
15999 REM rutina de fijacion de un centro y un eje de la transformacion
16000 TAG
16004 REM correcciones para centrar un caracter en un punto; la izquierda debe
ser 8 en modo 0 y 4 en modo 1
16005 correccionizquierda=2:correccionarriba=4
16010 MOVE xcentro-correccionizquierda,ycentro+correccionarriba:PRINT CHR$(129);
16100 WHILE respuesta$<>"x" AND respuesta$<>"y"
16110   respuesta$=LOWER$(INKEY$)
16115   xinc=0:yinc=0
16120   IF respuesta$="a" THEN yinc=2:respuesta$="":GOSUB 16500
16130   IF respuesta$="z" THEN yinc=-2:respuesta$="":GOSUB 16500
16140   IF respuesta$="," THEN xinc=-4:respuesta$="":GOSUB 16500
16150   IF respuesta$="." THEN xinc=4:respuesta$="":GOSUB 16500
16200 WEND
16220 MOVE xcentro-correccionizquierda,ycentro+correccionarriba:PRINT CHR$(129);
16230 RETURN
16499 REM borrar antiguo centro y dibujar el nue vo
16500 REM
16510 MOVE xcentro-correccionizquierda,yce ntro+correccionarriba:PRINT
CHR$(129);:xcentro=xcentro+xinc:ycentro=ycentro+yinc:MOVE xcentro-
correccionizquierda,ycentro+correccionarriba:PRINT CHR$(129);
16600 RETURN
```

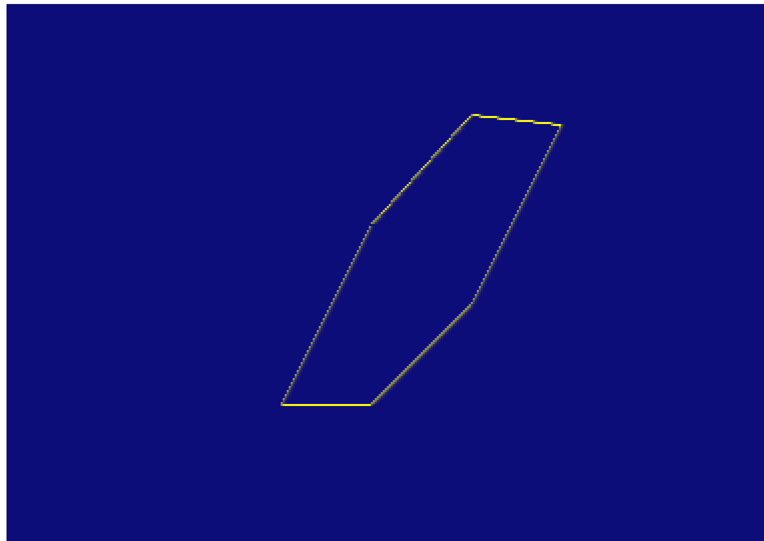


La simetría sobre rectas diferentes de los ejes es más delicada y requiere otras transformaciones.

Cizalladura

Se trata de un movimiento en el que el desplazamiento de cada punto depende de su distancia al eje del movimiento:

```
2999 REM 'c.' para cizallamiento
3080 IF respuesta$="c" THEN GOSUB 8000
7995 REM rutina de cizallamiento en la direccion de un eje
7996 REM el eje es vertical u horizontal
7997 REM pasando por el punto que se desee
7998 REM el punto se selecciona con a/z/,/.y se fija
7999 REM al pulsar 'x' o 'y' , lo que selecciona ademas el eje
8000 GOSUB 16000
8099 REM coeficientes de la transformacion
8100 xtrans1=1:ytrans2=1
8110 IF respuesta$="x" THEN ytrans1=1:xtrans2=0 ELSE xtrans2=1:ytrans1=0
8199 REM borra la figura anterior
8200 GOSUB 2000
8299 REM calcula la figura transformada
8300 GOSUB 12000
8399 REM dibujo de la figura transformada
8400 GOSUB 2000
8499 respuesta$=""
8500 RETURN
```



Ejercicios

1. Todas las transformaciones del programa se realizan sobre una figura extraída de un DATA. Añada una rutina al programa que le permita dibujar una figura y luego transformarla.
2. Modifique el programa para que sea posible que la figura inicial permanezca en la pantalla junto a la transformada. (Esto puede producir dibujos interesantes.)
3. Las transformaciones matriciales se pueden utilizar en la obtención de formas especiales, como las que obtuvimos en el capítulo 4 mediante rotación repetida y alargamiento. Escriba un programa que permita especificar una serie de transformaciones sucesivas a realizar con una figura. Haga que se pueda elegir el número de veces que se repetirá cada transformación. Tras introducirle estos datos, el ordenador deberá realizar las operaciones indicadas, dibujando las figuras que vayan resultando. Haga que cada figura se dibuje en un color, para obtener mejores efectos.
4. Los programas de CAD (Computer-Aided Design o "Diseño asistido por ordenador") contienen un menú de figuras diversas. Cuando el usuario elige una de ellas con el cursor, la figura queda pegada al cursor y se desplaza con él hasta que la figura se fija en el lugar deseado. Escriba un programa sencillo para CAD que permita simplificar el dibujo de una casa. Incluya como figuras varios tipos de puertas y ventanas. (Añada detalles que distingan cada tipo, pero no tantos como para que resulte demasiado lento el dibujo y borrado de las figuras.)

Técnicas de Programación de Gràficos en el Amstrad

La calidad de la presentación en la pantalla es fundamental para muchos programas. Este libro enseña a aprovechar plenamente las excelentes funciones gráficas del Amstrad. En los ejemplos se incluyen rutinas que el lector puede incorporar fácilmente a sus propios programas.

Entre los temas tratados se encuentran los juegos de tipo «máquinas tragaperras», la animación de figuras sencillas, realización de imágenes en color y su grabación en cinta, diseño de diagramas de barras y de sectores, cambios de escala y transformación de figuras y muchas otras aplicaciones interesantes. En todos los capítulos se dan sugerencias para ampliar y mejorar los programas de los ejemplos.

El autor

Wynford James se dedica a escribir material didáctico (incluidos programas) para una importante empresa de microordenadores. También ha trabajado como autor técnico para ICL. Ha sido profesor de matemáticas y ha participado en el desarrollo de estudios informáticos.

AMSTRAD
ESPAÑA
AVDA. DEL MEDITERRANEO, 9 - 28007 MADRID